<table>
<thead>
<tr>
<th>NAME:</th>
<th>DATE TAKEN</th>
<th>DATE RETURNED</th>
<th>SIGNATURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frank Togo</td>
<td>June 30/80</td>
<td>June 25/80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>July 14/80</td>
<td>July 14/80</td>
<td></td>
</tr>
</tbody>
</table>
AN IMPROVED THIRD NORMAL FORM FOR RELATIONAL DATA BASES

Tok-Wang Ling †
Frank Wm. Tompa
Tiko Kameda ‡‡

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

ABSTRACT

In this paper, we show that some Codd third normal form relations may contain "superfluous" attributes because the definitions of transitive dependency and prime attribute are inadequate when applied to sets of relations. To correct this, an improved third normal form is defined and an algorithm is given to construct a set of relations from a given set of functional dependencies in such a way that the superfluous attributes are guaranteed to be removed. This new normal form is compared with other existing definitions of third normal form, and the deletion normalization method proposed is shown to subsume the decomposition method of normalization.

Key Words and Phrases: data base design, relational schema, functional dependency, transitive dependency, prime attribute, third normal form, normalization, covering, reconstructibility.

CR Categories: 3.70, 4.33

This paper has been accepted for publication in Transactions on Database Systems.

July 12, 1980

† Current address: Department of Computer Science, Nanyang University, Upper Jurong Road, Singapore ‡‡ Department of Electrical Engineering
AN IMPROVED THIRD NORMAL FORM FOR RELATIONAL DATA BASES

Tok-Wang Ling †
Frank Wm. Tompa
Tiko Kameda ††

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

1. Introduction.

Several years ago the relational model was introduced in order that data base design could be grounded in a well-established mathematical discipline. The basic notion of a relation was augmented by the concepts of functional dependencies and normal forms in an attempt to provide integrity by reducing undesirable updating anomalies [Codd 71]. A particularly undesirable form of redundancy is the presence in a relation of an attribute whose value can always be derived from other attributes (perhaps using other relations) and whose value is not needed to derive other attributes’ values. Such an attribute will be called superfluous in that relation: a formal definition will be given in Section 4.

In this paper, we give examples to show that some Codd third normal form relations and some Codd’s normal form relations [Codd 74] may contain some superfluous attributes because the definitions of transitive dependency and prime attribute are inadequate when applied to such relations.

To correct this, we give new definitions to replace the usual definitions of these attributes. A normal form for a set of relations is then defined based on these new definitions. Since the old definitions are inadequate, all existing normalization methods need to be re-evaluated in the light of the new definitions. We present the detection of superfluous attributes, which are powers of the decomposition method, and we show the set of relations produced by the algorithm contains no superfluous attributes.

2. The relational model

A relational data base, consisting of several interrelated relations, was first introduced by Codd [Codd 70]. A relation is defined as follows: given sets of atomic (non-decomposable) elements $\text{DOM}_1, \text{DOM}_2, \ldots, \text{DOM}_n$ (not necessarily distinct), $\mathbf{T}$ is a first normal form relation (or simply relation) on these $n$ sets if it is a set of ordered $n$-tuples $(D_1, D_2, \ldots, D_n)$ such that $D_i$ belongs to $\text{DOM}_i$ for $i = 1, 2, \ldots, n$. Thus $\mathbf{T} \subseteq \text{DOM}_1 \times \text{DOM}_2 \times \ldots \times \text{DOM}_n$, where $\times$ denotes the Cartesian product. $\text{DOM}_1, \text{DOM}_2, \ldots, \text{DOM}_n$ are called the domains of $\mathbf{T}$. Rather than referencing each use of a domain by position number, each is assigned a unique role name, called an attribute of $\mathbf{T}$. For any tuple in $\mathbf{T}$, the value for the attribute named $B$ is referred to as a $B$-value, and, for a set of attributes $X = \{B_1, B_2, \ldots, B_p\}$, the tuple’s values for the attributes in $X$ is referred to as an $X$-value; the values of the other attributes in that tuple are said to be associated.

† Current address: Department of Computer Science, Nanyang University, Upper Jurong Road, Singapore 22
†† Department of Electrical Engineering

The work reported in this paper was partially supported by the Natural Sciences and Engineering Research Council of Canada under grants A9292 and A4315.
with that X-value. A set of attributes Y of T is said to be functionally dependent on a set of attributes X of T if each X-value in T has associated with it exactly one Y-value in T (at any time). This is denoted by \( X \rightarrow Y \) and is called a functional dependency of T; X and Y are termed the left and right sides of the dependency, respectively.

The relational algebra originally proposed by Codd includes several operations for manipulating relations. Of particular interest here are the operations of projection and (natural) join, defined

\[ T_1 \bowtie T_2 \text{ for some } T_1 \bowtie T_2 \text{ and } T_1 \bowtie T_2 = \{ \{ x \mid \exists y : (x, y) \in T_1 \times T_2 \} \} \]

where \( \bowtie \) denotes catenation (and possibly reordering of attributes).

It is important to realize that as data occurrences are inserted, deleted, and modified in a data base, the relations (that is, the sets of tuples) in that data base are altered. However, the relational algebra does not include facilities for altering a relation's set of attributes nor its set of functional dependencies. Thus these two sets are time-invariant properties associated with the relation scheme R which serves as a framework for a time-varying sequence of relations T. Henceforth, the notions of projection and join, when applied to relation schemes, refer to the corresponding relations when R remains fixed, and \( R \times R \) to mean the product relation.

dependencies \( F \), a functional dependency \( X \rightarrow BEF^+ \), where \( X \subseteq A \) and \( B \subseteq A \), is said to be a full dependency of \( R \) (or \( B \) is fully dependent on \( X \) under \( F \)) if there exists no proper subset \( X' \subseteq X \) such that \( X' \rightarrow BEF^+ \). Two sets of attributes \( X \subseteq A \) and \( Y \subseteq A \) are said to be functionally equivalent (or simply equivalent) if \( X \rightarrow Y \in F \) and \( Y \rightarrow X \in F \). \( X \) and \( Y \) are said to be properly functionally equivalent (or simply properly equivalent) if \( X \) and \( Y \) are equivalent and there exist no proper subsets \( X' \subseteq X \) and \( Y' \subseteq Y \) such that \( X' \rightarrow Y \in F \) or \( Y' \rightarrow X \in F \). An attribute \( B \) is said to be transitively dependent on \( X \subseteq A \) if there exists \( Y \subseteq A \) such that \( BEA \rightarrow Y \), \( X \rightarrow Y \in F \), \( Y \rightarrow BE \), and \( Y \rightarrow X \in F \).

For a relation scheme \( R \) having a set of attributes \( A \) and a set of functional dependencies \( F \), a set of attributes \( K \subseteq A \) is called a candidate key (or simply key) of \( R \) (or, colloquially, a key for \( A \)) if \( K \rightarrow A \in F \) and for all \( X \subseteq K \), \( X \rightarrow A \in F \). It is easy to prove that every relation has at least one key and that some may have more than one key. An attribute in \( A \) is called a prime attribute of \( R \) if it is contained in some key of \( R \). All other attributes in \( A \) are called non-prime attributes.
Codd recognized immediately that certain relation schemes may contain some redundancy. Consider, for example, the relation in Figure 1a which represents stock information for some hypothetical manufacturer:

<table>
<thead>
<tr>
<th>MODEL#</th>
<th>SERIAL#</th>
<th>PRICE</th>
<th>COLOUR</th>
<th>NAME</th>
<th>YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>342</td>
<td>13.25</td>
<td>blue</td>
<td>pot</td>
<td>1974</td>
</tr>
<tr>
<td>1234</td>
<td>347</td>
<td>13.25</td>
<td>red</td>
<td>pot</td>
<td>1974</td>
</tr>
<tr>
<td>1234</td>
<td>410</td>
<td>14.23</td>
<td>red</td>
<td>pan</td>
<td>1975</td>
</tr>
<tr>
<td>1465</td>
<td>347</td>
<td>9.45</td>
<td>black</td>
<td>pan</td>
<td>1974</td>
</tr>
<tr>
<td>1465</td>
<td>390</td>
<td>9.82</td>
<td>black</td>
<td>pan</td>
<td>1976</td>
</tr>
<tr>
<td>1465</td>
<td>392</td>
<td>9.82</td>
<td>red</td>
<td>pan</td>
<td>1976</td>
</tr>
<tr>
<td>1465</td>
<td>401</td>
<td>9.82</td>
<td>red</td>
<td>pan</td>
<td>1976</td>
</tr>
<tr>
<td>1465</td>
<td>409</td>
<td>9.82</td>
<td>blue</td>
<td>pan</td>
<td>1976</td>
</tr>
<tr>
<td>1623</td>
<td>311</td>
<td>22.34</td>
<td>blue</td>
<td>kettle</td>
<td>1973</td>
</tr>
<tr>
<td>1623</td>
<td>390</td>
<td>30.21</td>
<td>blue</td>
<td>kettle</td>
<td>1976</td>
</tr>
<tr>
<td>1623</td>
<td>410</td>
<td>28.55</td>
<td>black</td>
<td>kettle</td>
<td>1975</td>
</tr>
<tr>
<td>1623</td>
<td>423</td>
<td>28.55</td>
<td>black</td>
<td>kettle</td>
<td>1975</td>
</tr>
<tr>
<td>1623</td>
<td>428</td>
<td>28.55</td>
<td>blue</td>
<td>kettle</td>
<td>1975</td>
</tr>
<tr>
<td>1654</td>
<td>435</td>
<td>28.55</td>
<td>red</td>
<td>kettle</td>
<td>1975</td>
</tr>
</tbody>
</table>

Figure 1a. Stock inventory universal relation

<table>
<thead>
<tr>
<th>NAME</th>
<th>YEAR</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pot</td>
<td>1974</td>
<td>13.25</td>
</tr>
<tr>
<td>pot</td>
<td>1975</td>
<td>14.23</td>
</tr>
<tr>
<td>pan</td>
<td>1974</td>
<td>9.45</td>
</tr>
<tr>
<td>pan</td>
<td>1976</td>
<td>9.82</td>
</tr>
<tr>
<td>kettle</td>
<td>1973</td>
<td>22.34</td>
</tr>
<tr>
<td>kettle</td>
<td>1975</td>
<td>28.55</td>
</tr>
<tr>
<td>kettle</td>
<td>1976</td>
<td>30.21</td>
</tr>
</tbody>
</table>

Figure 1b. Stock inventory price relation

For each stock item, the model number, serial number, list price, colour, model name, and year of manufacture for an article are entered. The price and colour are unique for a given model number and serial number. If it is further assumed that the model name can be determined from the model number, the year of manufacture can be determined from the serial number, and the price can be determined from the model name and the year, then the set of functional dependencies is

\[
\{ \text{MODEL#}, \text{SERIAL#} \rightarrow \{ \text{PRICE}, \text{COLOUR} \}, \quad \text{MODEL#} \rightarrow \{ \text{NAME} \}, \\
\text{SERIAL#} \rightarrow \{ \text{YEAR} \}, \quad \text{NAME}, \text{YEAR} \rightarrow \{ \text{PRICE} \} \}
\]

If a new model for some year is announced but no items for that model and year are yet in stock (and therefore no model number and serial number are yet available), then the price information cannot be entered for that model (i.e., NAME) and year (the use of null or undefined values in other fields could cause problems [Osborn 77]). This is called the insertion anomaly. Now if the last item of stock for a particular model and year is sold and therefore a tuple with a \{NAME, YEAR\}-value that appeared in this tuple only is deleted, then the price information for this \{NAME, YEAR\}-value would be lost. This is called the deletion anomaly. If the \{PRICE\}-value for a \{NAME, YEAR\}-value were to be changed then that attribute's values for all tuples that have this given \{NAME, YEAR\}-value would also have to be changed to maintain the consistency of the data base. This is called the rewriting anomaly. Now suppose that the data base contains
another relation containing all the model name, year, and price information as in Figure 1b. In this case, the superfluous attribute PRICE could be removed from the universal stock relation scheme without losing any information from the database, whereas it would not be removable from the stock inventory price scheme without re-introducing anomalies.

One process that attempts to remove undesirable updating anomalies and redundant attributes from the relation schemes is called normalization, which was originally defined in two stages [Codd 71]. A (first normal form) relation scheme \( R \) is in second normal form if every non-prime attribute of \( R \) is fully dependent on each key of \( R \). A relation scheme \( R \) is in Codd third normal form if it is in second normal form and each non-prime attribute of \( R \) is not transitively dependent on every key of \( R \). For example, the set of relations in third normal form in Figure 2 maintains the same data as the stock inventory depicted in Figure 1.

<table>
<thead>
<tr>
<th>MODEL #</th>
<th>SERIAL #</th>
<th>PRICE</th>
<th>COLOUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>342</td>
<td>13.25</td>
<td>blue</td>
</tr>
<tr>
<td>1234</td>
<td>347</td>
<td>13.25</td>
<td>red</td>
</tr>
<tr>
<td>1234</td>
<td>410</td>
<td>14.23</td>
<td>red</td>
</tr>
<tr>
<td>1465</td>
<td>347</td>
<td>9.45</td>
<td>black</td>
</tr>
<tr>
<td>1465</td>
<td>390</td>
<td>9.82</td>
<td>black</td>
</tr>
<tr>
<td>1465</td>
<td>392</td>
<td>9.82</td>
<td>red</td>
</tr>
<tr>
<td>1465</td>
<td>401</td>
<td>9.82</td>
<td>red</td>
</tr>
<tr>
<td>1465</td>
<td>409</td>
<td>9.82</td>
<td>blue</td>
</tr>
<tr>
<td>1623</td>
<td>311</td>
<td>22.34</td>
<td>blue</td>
</tr>
<tr>
<td>1623</td>
<td>390</td>
<td>30.21</td>
<td>blue</td>
</tr>
<tr>
<td>1623</td>
<td>410</td>
<td>28.55</td>
<td>black</td>
</tr>
<tr>
<td>1623</td>
<td>423</td>
<td>28.55</td>
<td>black</td>
</tr>
<tr>
<td>1623</td>
<td>428</td>
<td>28.55</td>
<td>blue</td>
</tr>
<tr>
<td>1654</td>
<td>435</td>
<td>28.55</td>
<td>red</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SERIAL #</th>
<th>YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>311</td>
<td>1973</td>
</tr>
<tr>
<td>342</td>
<td>1974</td>
</tr>
<tr>
<td>347</td>
<td>1974</td>
</tr>
<tr>
<td>390</td>
<td>1976</td>
</tr>
<tr>
<td>392</td>
<td>1976</td>
</tr>
<tr>
<td>401</td>
<td>1976</td>
</tr>
<tr>
<td>409</td>
<td>1976</td>
</tr>
<tr>
<td>410</td>
<td>1975</td>
</tr>
<tr>
<td>423</td>
<td>1975</td>
</tr>
<tr>
<td>428</td>
<td>1975</td>
</tr>
<tr>
<td>435</td>
<td>1975</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>YEAR</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pot</td>
<td>1974</td>
<td>13.25</td>
</tr>
<tr>
<td>pot</td>
<td>1975</td>
<td>14.23</td>
</tr>
<tr>
<td>pan</td>
<td>1974</td>
<td>9.45</td>
</tr>
<tr>
<td>pan</td>
<td>1976</td>
<td>9.82</td>
</tr>
<tr>
<td>kettle</td>
<td>1973</td>
<td>22.34</td>
</tr>
<tr>
<td>kettle</td>
<td>1975</td>
<td>28.55</td>
</tr>
<tr>
<td>kettle</td>
<td>1976</td>
<td>30.21</td>
</tr>
</tbody>
</table>

Figure 2. Stock inventory normalized relations

THEOREM 1. A relation scheme \( R \) is in Codd third normal form if and only if each non-prime attribute is not transitively dependent on an arbitrarily chosen key of \( R \).

PROOF: The proof is based on the following two lemmas which show that a non-full or transitive dependency of an attribute on any one key of \( R \) implies the transitive dependency of that attribute on all keys of \( R \).

LEMMA 1.1 Let \( R \) be a relation scheme consisting of a set of attributes \( A \) and a set of functional dependencies \( F \) and let \( B \in A \). If there exists a key \( K \) of \( R \) with \( B \in K \) and \( K \to B \) is not a full dependency, then \( B \) is transitively dependent on all keys of \( R \).

PROOF: Since \( B \in K \) and \( K \to B \) is not a full dependency, therefore there exists a proper subset \( X \subset K \) such that \( B \in X \) and \( X \to B \in F^+ \). Since \( K \) is a key and \( X \) is a proper subset of \( K \), \( X \to K \in F^+ \). Now for any key \( K' \) of \( R \), \( K' \to X \in F^+ \), \( X \to K \in F^+ \), \( X \to B \in F^+ \), and \( B \in X \). Hence \( B \) is transitively dependent on \( K \), which proves the lemma.
**Lemma 1.2** Let \( R \) be a relation scheme consisting of a set of attributes \( A \) and a set of functional dependencies \( F \) and let \( B \subseteq A \). If there exists a key \( K \) of \( R \) such that \( B \) is transitively dependent on \( K \), then \( B \) is also transitively dependent on all keys of \( R \).

**Proof:** Let \( K' \) be any other key of \( R \). By definition, \( K' \rightarrow B \in F^+ \). Now if \( B \) is transitively dependent on \( K \), then there exists a set of attributes \( X \subseteq A \) such that \( K' \rightarrow X \in F^+ \), \( X \rightarrow K \in F^+ \), \( X \rightarrow B \in F^+ \), and \( B \in X \). Hence \( K' \rightarrow X \in F^+ \), \( X \rightarrow K \in F^+ \), \( X \rightarrow B \in F^+ \), and \( B \in X \). Thus \( B \) is transitively dependent on \( K' \), which proves the lemma.

Closely related to closure is the notion of derivability [Lucchesi 78], as follows: a set of attributes \( Y \) is derivable from a set of attributes \( X \) using the set of functional dependencies \( F \) if there exists a sequence of attribute sets (called a derivation of \( Y \) from \( X \)) \( <X_0,X_1,X_2,\ldots,X_n> \) for \( n \geq 0 \) such that \( X_0 = X \), \( Y \subseteq X_n \), and (unless \( n = 0 \)) for \( i \) in the range 1 to \( n \) there exists a functional dependency \( X_{i-1} \rightarrow X_i \in F \) such that \( V \subseteq X_{i-1}, W \subseteq X_{i-1}, \) and \( Y \subseteq X_n \).

**Theorem 2.** [Lucchesi 78, Beeri 79] Given a set of attributes \( A \), \( X \subseteq A \) and \( Y \subseteq A \), and a set of functional dependencies \( F \) defined on subsets of \( A \), \( Y \) is derivable from \( X \) using \( F \) if and only if \( X \rightarrow Y \in F^+ \). Furthermore, since \( B_{i-1} \subseteq B_i \) and \( B_{i-1} \not\subseteq B_i \), derivability can be decided in \( O(|F| + |A|) \) time.\(^*\)

Instead of considering the derivation of a particular set of attributes \( Y \), it is sometimes convenient to find the set of all attributes derivable from \( X \) using \( F \). A maximal derivation from a set of attributes \( X \) using a set of functional dependencies \( F \) is a sequence of attribute sets \( <X_0,X_1,X_2,\ldots,X_n> \) for \( n \geq 0 \) such that \( X_0 = X \), \( X_i \subseteq X_{i+1} \), and \( Y \subseteq X_n \). Lucchesi and Osborn have shown that the terminal set in a maximal derivation from \( X \) using \( F \) is independent of the particular derivation: henceforth \( X_n \) will be called the closure of \( X \) relative to \( F \).

3. The problem of superfluous attributes

Given \( A \), a set of attributes, and \( F \), a set of functional dependencies among subsets of \( A \), it is desirable to describe a set of relation schemes, henceforth called a relational scheme, \( R = \{R_1, \ldots, R_n\} \) such that the following three properties hold:

1. The relationships among the data values to be stored using \( R \) are equivalent to those that would be stored using a single relation scheme \( R_0 \) involving all of \( A \). That is, at all times, \( R_i = R_0[A_i] \) where \( A_i \) is the set of attributes in \( R_i \), and \( R_1 \cdot R_2 \cdot \ldots \cdot R_n = R_0 \).

2. The verification that a set of relations described by \( R \) (i.e., relational instances for \( R_1, \ldots, R_n \)) conforms to all functional dependencies in \( F \) requires only the examination of relations corresponding to \( R_1, \ldots, R_n \) individually. Furthermore, the only functional dependencies that need be examined are those for which the left side contains a key of some \( R_i \). It is required that \( \bigcup \{X \rightarrow Y \mid X \subseteq A_i \text{ and } X \rightarrow Y \in F^+ \} = F^+ \).

3. Each relation scheme is free of redundant attributes, that is, those whose presence is not required for maintaining the other two properties. It should be noted here that the redundancy considered is with respect to given sets of attributes and functional dependencies: the redundancy under consideration is thus not that which may arise from other time-invariant properties of the data base (see, for example, [Delobel 78]).

These goals have been defended elsewhere (see, for example, [Rissanen 77, Beeri 78, Biskup 79]) and have been termed reconstructability (or losslessness), covering, and normalization, respectively.

Several characterizations for the elements of \( R \) have been given since the introduction of the relational model, but none have satisfactorily met the requirement of normalization in that redundant attributes are sometimes permitted. For the remainder of this section, we will denote the
LEMMA 1.2 Let $R$ be a relation scheme consisting of a set of attributes $A$ and a set of functional dependencies $F$ and let $B \subseteq A$. If there exists a key $K$ of $R$ such that $B$ is transitively dependent on $K$, then $B$ is also transitively dependent on all keys of $R$.

PROOF: Let $K'$ be any other key of $R$. By definition, $K' \rightarrow B \in F^+$. Now if $B$ is transitively dependent on $K$, then there exists a set of attributes $X \subseteq A$ such that $K \rightarrow X \in F^+$, $X \rightarrow K \in F^+$, $X \rightarrow B \in F^+$, and $B \in E$. Hence $K' \rightarrow X \in F^+$, $X \rightarrow K \in F^+$, $X \rightarrow B \in F^+$, and $B \in E$. Thus $B$ is transitively dependent on $K'$, which proves the lemma.

Closely related to closure is the notion of derivability [Lucchesi 78], as follows: an attribute $Y$ is derivable from a set of attributes $X$ using the set of functional dependencies $F$ if there exists a sequence of attribute sets (called a derivation of $Y$ from $X$) $< X_0, X_1, X_2, \ldots, X_n >$ for $n \geq 0$ such that $X = X_0$, $Y \subseteq X_n$, and (unless $n = 0$) for $i$ in the range 1 to $n$ there exists a functional dependency $V \rightarrow W \in F$ such that $V \subseteq X_{i-1}$, $W \subseteq X_{i-1}$, and $X_i = X_{i-1} \cup W$.

THEOREM 2. [Lucchesi 78, Beeri 79] Given a set of attributes $A$, $X \subseteq A$, and $Y \subseteq A$, and a set of functional dependencies $F$ defined on subsets of $A$, $Y$ is derivable from $X$ using $F$ if and only if $X \rightarrow Y \in F^+$. Furthermore, since $B_i \subseteq C \subseteq B_i$ and $B_i \neq B_i$, derivability can be decided in $O(|F| |A|)$ time.‡

Instead of considering the derivation of a particular set of attributes $Y$, it is sometimes convenient to find the set of all attributes derivable from $X$ using $F$. A maximal derivation from a set of attributes $X$ using a set of functional dependencies $F$ is a sequence of attribute sets $< X_0, X_1, \ldots, X_n >$ for $n \geq 0$ such that $X = X_0$, for $i$ in the range 1 to $n$ there exists a functional dependency $V \rightarrow W \in F$ such that $V \subseteq X_{i-1}$ and $W \subseteq X_{i-1}$ and $X_i = X_{i-1} \cup W$, and there is no functional dependency $V \rightarrow W \in F$ such that $V \subseteq X_n$ and $W \subseteq X_n$. Lucchesi and Osborn have shown that the terminal set in a maximal derivation from $X$ using $F$ is independent of the particular derivation; henceforth $X_n$ will be called the closure of $X$ relative to $F$.

3. The problem of superfluous attributes

Given $A$, a set of attributes, and $F$, a set of functional dependencies among subsets of $A$, it is desirable to describe a set of relation schemes, henceforth called a relational scheme,* $R = [R_1, \ldots, R_n]$ such that the following three properties hold:

1. The relationships among the data values to be stored using $R$ are equivalent to those that would be stored using a single relation scheme $R_0$ involving all of $A$. That is, at all times $R_j = R_0[A_j]$ where $A_j$ is the set of attributes in $R_j$, and $R_1 \ast R_2 \ast \cdots \ast R_n = R_0$.

2. The verification that a set of relations described by $R$ (i.e., relational instances for $R_1, \ldots, R_n$) conforms to all functional dependencies in $F$ requires only the examination of relations corresponding to $R_1, \ldots, R_n$ individually. Furthermore, the only functional dependencies that need be examined are ones for which the left side contains a key of some $R_j$. It is required that $(\bigcup \{X \rightarrow A_i, X \subseteq A_j \text{ and } X \rightarrow A_i \in F^+\})^+ = F^+$.

3. Each relation scheme is free of redundant attributes, that is, those whose presence is not required for maintaining the other two properties. It should be noted here that the redundancy considered is with respect to given sets of attributes and functional dependencies only; the redundancy under consideration is thus that which may arise from other functionally invariant properties of the data base (see, for example, [Delobel 78]).

These goals have been defended elsewhere (see, for example, [Rissanen 77, Beeri 78, Biskup 79]) and have been termed reconstructibility (or losslessness), covering, and normalization, respectively.

Several characteristics for the elements of $R$ have been given since the introduction of the relational model, but none have satisfactorily met the requirement of normalization in the sense that redundancies are sometimes permitted. For the remainder of this section, we will denote...
Example 1. Let \( A = ABCDEFG \) and \( F = \{ AB \rightarrow CD, A \rightarrow F, B \rightarrow F, EF \rightarrow C \} \) and consider the relational schema \( R = \{ R_1, R_2, R_3, R_4 \} \) where \( R_1 \) has attributes \( ABCD \) and key \( AB \); \( R_2 \) has attributes \( BC \) and key \( B \); \( R_3 \) has attributes \( BF \) and key \( B \); and \( R_4 \) has attributes \( EF \) and key \( EF \). The figure models the relations depicted in Figure 2, where \( A \) is the model number, \( B \) the part number, \( C \) the colour, \( D \) the model name, and \( F \) is the year of manufacture. Each \( R_i \) has the properties of reconstructibility and covering, and that each relation in \( R \) is in 3NF. However, \( X \rightarrow C \in F^+ \) and \( X \rightarrow AB \in F^+ \), hence \( C \) is not transitively dependent on \( AB \) in \( R \). Therefore, \( AB \rightarrow D \) is also not a transitive dependency in \( R \). Thus \( R_1 \) is in 3NF and \( R \) contains \( R_2, R_3, \) and \( R_4 \). Now consider an instance of \( R \) containing tuples \( (A_1, B_1, C_1, D_1, F_1) \) for \( R_1 \), \( (B_1, E_1) \) for \( R_2 \), \( (B_1, F_1) \) for \( R_3 \), and \( (E_1, F_1, C_2) \) for \( R_4 \). Here the \( AB \)-value \( A_1B_1 \) can code the \( C \)-value \( C_2 \) by using relations other than the one corresponding to \( R \). If \( C_1 \neq C_2 \), then even if \( R_1 \) satisfies the functional dependency \( AB \rightarrow C \), the data base will be inconsistent. Similarly, if the data base is consistent, the \( C \)-value for some tuple for \( R_1 \) cannot be altered without checking all other relations that the data base will not become inconsistent; that is, this normalization has not eliminated the potential for updating anomalies. It is easy to show that although \( AB \rightarrow C \) is not a transitive dependency in \( R \), \( C \) is a superfluous attribute, i.e., it can be deleted from the relation \( R_1 \) while still preserving the functional dependency \( AB \rightarrow C \) in \( R \).

Example 2. Let \( A = ABCDEFG \) and \( F = \{ AD \rightarrow B, B \rightarrow C, C \rightarrow D, AB \rightarrow E, C \rightarrow F \} \). Consider the relational schema \( R = \{ R_1, R_2, R_3 \} \) where \( R_1 \) has attributes \( ABCDEF \) and key \( AB \); \( R_2 \) has attributes \( BC \) and key \( B \); and \( R_1 \) has attributes \( CD \) and key \( C \). Again, \( R \) has the properties of reconstructibility and covering, and each relation in \( R \) is in 3NF. In this case, the attribute \( C \) is superfluous in \( R_1 \) and, although it is a prime attribute, it need not be considered for transitive dependencies when constructing 3NF forms. Hence \( C \) (and \( F \)) should be dropped from \( R_1 \) while still preserving all functional dependencies in \( R \).

From these two examples it is clear that the definitions of transitive dependence and superfluous attributes as applied to defining 3NF are inadequate for describing relations that involve "superfluous" attributes. Since the definitions are inadequate, any normalization that is based on them may also be inadequate. For example, Bernstein's method [Bernstein 71] for, in fact, produces sets of relation schemes as in Example 2; that is, the method may produce schemes that are free of "superfluous" attributes.

Furthermore, the definition of Boyce-Codd normal form is inadequate for describing relations that involve "superfluous" attributes. For example, Bernstein's method [Bernstein 71] for producing a set of functional dependencies \( F \) there may not exist a relational schema that covers \( F \) [Codd 71]. Thus the definition does not ensure that Boyce-Codd normalization is a necessary and sufficient condition for eliminating all first normal forms, reconstructibility, covering all functional dependencies.
4. An improved third normal form

Because normalization should always be achieved in addition to covering and reconstructibility, the properties of a normal form will henceforth be discussed solely in the context of a relational schema that has the other two properties. Thus given $\mathcal{A}$, a set of attributes, and $\mathcal{F}$, a set of functional dependencies involving subsets of $\mathcal{A}$, it is first desirable to obtain a relational schema that is satisfactory except for normalization. The following algorithm is based upon Bernstein's synthesis algorithm [Bernstein 76, p. 293]:

Preparatory algorithm.

Input. $\mathcal{A}$, a set of attributes, and $\mathcal{F}$, a set of functional dependencies on $\mathcal{A}$.

1. (Remove extraneous attributes and dependencies.) Eliminate from both sides of each functional dependency in $\mathcal{F}$ all attributes whose elimination leaves a set of functional dependencies having a closure equal to $\mathcal{F}^+$. Next eliminate from that modified set all functional dependencies whose right side is the empty set of attributes. Let $\mathcal{F}_1$ be the resulting set.

2. (Partition the functional dependencies.) Partition $\mathcal{F}_1$ into a set of classes $\mathcal{C}$ such that all the functional dependencies in each class have properly equivalent left sides, that is $V_1 \rightarrow W_1$ and $V_2 \rightarrow W_2$ are in the same class if and only if $V_1 \rightarrow W_1 \in \mathcal{F}^+$ and $V_2 \rightarrow W_2 \in \mathcal{F}^+$.

3. (Construct relation schemes.) For each class in $\mathcal{C}$, construct a relation scheme $\mathcal{R}_i$ consisting of all attributes $A_j$ appearing in that class. Let $\mathcal{R}$ be the set resulting from these constructions.

4. (Augment the relational schema, if necessary.) If for each relation scheme in $\mathcal{R}$, the set of attributes $A_i$ in that class is such that $A_i \rightarrow A \in \mathcal{F}^+$, then construct $A'$ to be a minimal set of attributes in $\mathcal{A}$ such that $A' \rightarrow A \in \mathcal{F}^+$ and augment $\mathcal{R}$ by one relation scheme whose attribute set is $A'$.

Output. $\mathcal{R}$, the preparatory relational schema.

It is simple to show that the set of attributes constituting the left side of a functional dependency in $\mathcal{F}_1$ is a key of the relation scheme having that dependency and constructed in Step 3. Bernstein has called each such set of attributes a synthesized key. Furthermore, if an additional relation scheme is introduced into $\mathcal{R}$ in Step 4, its synthesized key consists of all its attributes. To represent the dependencies contained in $\mathcal{F}$, the set $\mathcal{K}_f$ of synthesized keys will be recorded as part of the relation schemes in $\mathcal{R}$. In the rest of this paper, the phrase "Let $\mathcal{R}$ be a preparatory relational schema ...." is shorthand for "Given a set of attributes $\mathcal{A}$ and a set of functional dependencies $\mathcal{F}$ defined on subsets of $\mathcal{A}$, let $\mathcal{R}$ be a preparatory relational schema consisting of relation schemes $\mathcal{R}_i$, each having a set of attributes $\mathcal{A}_i$ and a set $\mathcal{K}_i$ of synthesized keys ...."; the notation used in examples will be $\mathcal{R} = \langle \mathcal{R}_1, \mathcal{A}_1, \mathcal{K}_1 >, \ldots, \mathcal{R}_n, \mathcal{A}_n, \mathcal{K}_n \rangle$.

Let $G_j$ be a set of synthesized functional dependencies in the relation scheme $\mathcal{R}_j$, that is $\mathcal{R}_j = G_j \cup \mathcal{K}_j$. For $G = \bigcup G_j$, $G^+ = F^+$ [Bernstein 76]; that is, $G$ covers all functional dependencies, as described in Section 3. Osborn has proven that in the presence of $\mathcal{R}$, $\mathcal{R}$ has the desired property of reconstructibility if one of the relation schemes in $\mathcal{R}$ is a key $\mathcal{K}$ such that $\mathcal{R} \rightarrow A \in \mathcal{F}^+$ [Osborn 77, Biskup 79]; this is guaranteed in Step 4. Thus Osborn would classify the elements of $\mathcal{R}$ as independent components [Rissanen 77, and Beeri, et al. classify $\mathcal{R}$ as a "redundancy-free representation" for $\mathcal{A}$ and $\mathcal{F}$ [Beeri 78].

Beeri and Bernstein have shown that Steps 1-3 can be computed in time proportional to the sum of the lengths of the input [Beeri 79]. Since the last step is similar to repeating Steps 1-3 in reverse order, and the output is the same, the preparatory algorithm runs in time $O(|\mathcal{F}| + 3|\mathcal{A}|)$.

Note: One or both of the proof steps are omitted from this algorithm: Osborn's non-redundancy theorem [Osborn 77, Biskup 79] is used, which it is shown that the algorithm satisfies. Given a minimal set of $\mathcal{A}$, the non-redundancy theorem [Bernstein 76, Osborn 77]. Given a functional schema.
the preparatory algorithm for given sets of attributes and functional dependencies, the normalization procedure proposed here will remove attributes from individual schemes and adjust the set of synthesized keys. For simplicity, any such derived relational schema will also be called a preparatory relational schema as long it maintains the properties of covering and reconstructibility.

The object of normalization is to remove unnecessary redundancy from a collection of relations. In particular, with respect to a relational schema \( R \), an attribute \( B \) is superfluous in a relation scheme \( R_i \) if its removal from \( R_i \) does not affect covering nor reconstructibility: that is, all data relationships stored in an instance of \( R \) can be reconstructed without reference to the attribute \( B \in R_i \). A more precise definition is given below.

Let \( R \) be a preparatory relational schema including \( R_i \), and let \( B \) be an attribute in \( A_j \). The functional dependencies that do not involve \( B \) in \( R_i \) may be defined as follows:

\[
D_i(B) = \bigcup_{j \neq i} \{ X \rightarrow A_j | X \subseteq A_j, X \rightarrow A_j \in F_i^+ \} \quad \text{and for no } X \subset X, X \rightarrow A_j \in F_i^+
\]

\[
\bigcup \{ X \rightarrow A_j | X \subseteq A_j, X \rightarrow A_j \in F_i^+ \}
\]

It is important to realize that \( D_i(B) \) is defined in terms of all keys for all relation schemes in \( R \) (denoted by the union of terms), not only keys synthesized by the preparatory algorithm. Thus \( B \) is superfluous in \( R_i \) if both of the following conditions hold:

1. (covering condition): the set of dependencies excluding those involving \( B \) in \( R_i \) covers \( F_i \); that is, \( D_i(B)^+ = F_i^+ \)

2. (reconstructibility condition): a key of \( R_i \) (see Section 3) is contained in some relation without involving \( B \) in \( R_i \); that is, \( A_j \rightarrow A \in F_i^+ \) for some \( j \neq i \) or \( A_j \rightarrow B \rightarrow A \in F_i^+ \).

Any algorithm that detects superfluous attributes by applying a straightforward implementation of these conditions requires the calculation of \( D_i(B)^+ \) for each possible value of \( i \) and \( B \), which in turn requires that all keys of all relations be found. Because the number of keys can be exponential in \( |A| \) and \( |F| \) [Yu 76, Demetrovics 78], an algorithm used in practice must avoid calculating all keys. Thus rather than implementing the conditions as above, alternative definitions, less intuitive but more practical, will be given first.

Since the preparatory algorithm synthesizes only a polynomial number of keys (in terms of \( |A| \) and \( |F| \)), it would be convenient to be able to ignore all keys that are not synthesized. As a parallel to \( D_i(B) \), let \( G_i(B) \) be the set of all synthesized dependencies that do not involve \( B \) in \( R_i \), that is,

\[
G_i(B) = \bigcup_{j \neq i} \{ K \rightarrow A_j | K \subseteq A_j, K \in K_i \} \quad \text{and for no } X \subset X, X \rightarrow A_j \in F_i^+
\]

An example will illustrate the difference between \( D_i(B) \) and \( G_i(B) \):

**Example 3.** Let \( A = ABCDE \) and \( F = \{ A \rightarrow B, B \rightarrow A, AC \rightarrow DE, BD \rightarrow C \} \). For the preparatory relational schema \( R = \{ R_1 \subseteq \{ A, B \}, R_2 \subseteq \{ ABCDE \} \} \), it can be seen that \( D_1(B) = \{ A \rightarrow B, B \rightarrow A, AC \rightarrow DE, AD \rightarrow CE \} \) and \( G_1(B) = \{ A \rightarrow B, B \rightarrow A, AC \rightarrow DE \} \). Notice that \( G_1(B)^+ \) is in \( D_1(B)^+ \) but not in \( G_1(B)^+ \); without the recognized of \( AD \) as a (non-)key of \( R_3 \), \( R \) would not be seen to be superfluous in \( R \).
R. (An attribute that does not meet the conditions for the second definition is said to be essential.)

Thus a non-prime attribute is obviously non-essential, and a transitive dependency will be shown to imply an attribute's transitivity, these facts which will be used in the proof of Theorem 4.

Together, the definitions of prime and non-prime attributes, as follows:

THEOREM 3. Let \( R \) be a relation. If an attribute B is essential in \( R \), and let \( B' \) be an attribute in \( A_j \). The attribute \( B \) is also non-essential in \( R_j \).

PROOF: Assume that \( B \) is essential in \( R \), then the relation \( R \) was introduced into \( R_j \). But then the restored dependency, \( \text{K} \rightarrow \text{B} \) was introduced into \( R_j \). Therefore, it must be the case that \( \text{K} \rightarrow \text{B} \) in \( R_j \), such that \( \text{K} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \), and thus \( \text{K} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \) is a synthesized dependency by hypothesis, since \( \text{B} \) is non-essential in \( R_j \). Since \( \text{B} \) is non-essential, \( \text{B} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \) for some \( j \).

Case: \( \text{B} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \), otherwise, since \( \text{B} \) is non-essential in \( R_j \), \( \text{K} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \) and \( \text{B} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \). Thus \( \text{B} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \). This completes the proof for this case.

LEMMA 3.4. Let \( R \) be a relation including \( R \), and let \( B \) be an attribute in \( R \). Then let \( \text{K}, \text{B} \in \text{X}_j \) such that \( \text{B} \in \text{K}, \text{K} 

PROOF: Assume that \( \text{K} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \) such that \( \text{B} \in \text{K}, \text{B} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \) but \( \text{K} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \rightarrow \ldots \) \( \text{B} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \). Let \( \text{K} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \) be a maximal derivation from \( K \) using \( \text{G}_j \). By assumption, \( \text{K} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \). However, \( \text{A}_j \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \). Since \( \text{R} \) has the property that \( \text{A} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \), it is shown that \( \text{K} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \), and thus \( \text{K} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \). Thus \( \text{K} \rightarrow \text{B} \rightarrow \text{B} \rightarrow \ldots \rightarrow \ldots \). This completes the proof for this case.
restorable in \( R_i \).

**Lemma 3.2.** Let \( R \) be a preparatory relational schema including \( R_i \), and let \( B \) be an attribute in \( A_j \). If for every key \( K \in K_i \) such that \( B \in K \), \( K \rightarrow A_j \in D_i(B)^+ \), then \( B \) is non-essential in \( R_i \).

**Proof:** Assume that for some \( K \in K_i \) such that \( B \in K \), \( K \rightarrow A_j \in D_i(B)^+ \). Let \( \langle X_0, X_1, \ldots, X_n \rangle \) be a maximal derivation from \( K \) using \( G_i(B) \). If \( A_j \not\subseteq X_n \), then the following argument shows that there must be a key for \( A_j \) in \( X_n \) and \( B \). If all keys for \( A_j \) contain \( B \), then, by definition, \( G_i(B) = G_i(G) \). Thus \( K \rightarrow A_j \in (G - G_i)^+ \), which can only result from a given functional dependency \( f : K \rightarrow Z \) being redundant, that is, \( f \in (F_f)^+ \). However, each such functional dependency would have been removed in Step 1 of the preparatory algorithm and \( R_i \) would not have been created, thus proving that if \( A_j \not\subseteq X_n \), then \( B \) is non-essential in \( R_i \).

If \( A_j \not\subseteq X_n \), then let \( \langle X_n, X_{n+1}, \ldots, X_m \rangle \) be a derivation of \( A_j \) from \( X_n \) using \( D_i(B) \), and let \( V \) and \( W \) be as defined in the proof of Lemma 3.1. Since, by the same argument, \( j = i \) and thus \( V \) is a key of \( R_i \), and since \( V \rightarrow W \in D_i(B) \) implies \( B \in V \), the closure of \( K \) relative to \( G_i(B) \) has been shown to contain a key for \( A_j \) which does not contain \( B \), and thus \( B \) is non-essential in \( R_i \).

Finally, using the definitions of restorable and non-essential, a characterization for normalization can be defined in a manner similar to the statement of Theorem 1 for Codd third normal form:

A relation scheme \( R_i \) in a preparatory relational schema \( R \) is in improved third normal form if each non-essential attribute is not restorable in \( R_i \).

Restorability in \( R_i \) indicates a form of "implicit" or "indirect" dependency on an arbitrarily chosen key that does not contain \( B \) [cf. Ling 78, Maier 79]; thus this definition is an exact analog for Theorem 1.

In re-examining the examples given in Section 3, it can be seen that the definition of improved third normal form captures the desired notion of non-redundancy.

**Example 1.** Let \( A = ABCDEF \) and \( F = \{ AB \rightarrow CD, A \rightarrow E, B \rightarrow F, EF \rightarrow C \} \). The preparatory algorithm will yield the relational schema \( R_i = \{ R_1, ABD, AB, AE, A, R_2, B, BF, B \} \). Since \( C \) is an extraneous attribute in \( AB \rightarrow CD \). It can be seen that there are no attributes that are both non-essential and restorable in any of the relation schemas, which are therefore all in improved third normal form.

**Example 2.** Let \( A = ABCDEF \) and \( F = \{ AD \rightarrow B, B \rightarrow C, E \rightarrow D \}, ABC \rightarrow E \). For the relational schema \( R = \{ R_1, C \} \), it can be seen that the attribute \( C \) is non-essential and restorable in \( R_i \) using \( G_i(B) = \{ B \rightarrow C, C \rightarrow D, ABC \rightarrow DEF, AD \rightarrow BEF \} \). ABCDEF is derivable from the only synthesized key involving \( C \) (i.e., \( AC \)) and \( C \) is derivable from \( AB \), a key not containing \( C \). Removing \( C \) from \( R_i \) leaves a relational schema each member of which is in improved third normal form.

**Theorem 4.** Let \( R \) be a preparatory relational schema including \( R_i \). If \( R_i \) is in improved third normal form, then it is also in Codd third normal form.

**Proof:** The following lemma shows that transitive dependencies for non-prime attributes result in those attributes being restorable. Together with the observation that all non-prime attributes are non-essential (since a non-prime attribute \( B \) is in no \( K \in K_i \)), the conditions for Theorem 1 necessarily occur in improved third normal form schemes, thus proving this theorem.

**Lemma 4.1.** Let \( R \) be a preparatory relational schema including \( R_i \), and let \( B \) be a non-prime attribute in \( A_j \). If \( B \) is transitively dependent on \( K \), a key of \( R_i \), it is restorable in \( R_i \).
**Proof:** If \( B \) is transitively dependent on \( K \), then for some \( X \) contained in \( A_f \), \( B \in X \). \( X \rightarrow X \in F^+ \), \( X \rightarrow K \in E^+ \), and \( X \rightarrow B \in E^+ \). Since \( B \) is non-prime, \( B \wedge K \) and therefore clearly \( K \notin \{A_f\} \). Since \( B \wedge K \wedge X \), \( X \rightarrow X \in G_f^-(B) \) and therefore \( K \rightarrow X \in G_f^-(B) \). Because \( X \rightarrow K \in E^+ \) and \( X \rightarrow B \in E^+ \), \( B \) is derivable from \( X \) using \( F \) whereas \( K \) is not. Thus, because the left side of each functional dependency used in the derivation \( X \rightarrow B \) cannot be properly equivalent to \( K \), each such dependency is placed by the preparatory algorithm in some class distinct from the one resulting in the construction of \( R_i \). Therefore \( B \) is derivable from \( X \) using \( G \rightarrow G_f^-(B) \); that is, \( X \rightarrow B \in G_f^-(B) \). Hence, by transitivity, \( K \rightarrow B \in G_f^-(B) \). For each key \( K' \) in \( K \), such that \( B \wedge K' \), \( K' \rightarrow K \in G_f^-(B) \); thus \( K' \rightarrow B \in G_f^-(B) \). Therefore \( B \) is restorable in \( R_i \).

It should be noted that Theorem 4 does not claim a necessary, but rather a sufficient, condition for Codd third normal form. Together with Theorems 3 and 4, the examples show that improved third normal form is superior to Codd third normal form in removing superfluous attributes.

An efficient deletion normalization algorithm can be derived by starting with the preparatory relational schema and repeatedly removing superfluous attributes. Since the result of each removal is again a preparatory relational schema, eventually such a normalization algorithm will result in a preparatory set in which there are no superfluous attributes; that is, the result will be a relational schema in improved third normal form.

Before describing this algorithm formally, it is convenient to give a formal description of an algorithm that indicates whether or not a given attribute is superfluous in a relation scheme. The algorithm determines the restorability of an attribute by appealing to the following lemma:

**Lemma.** Let \( R \) be a preparatory relational schema including \( R_i \), and let \( B \) be an attribute in \( A_f \) and \( K \) be a synthesized key with \( B \wedge K \). If \( K \rightarrow B \in G_f^-(B) \), then for each key \( K' \) in \( K \), such that \( B \wedge K' \), \( K' \rightarrow B \in G_f^-(B) \).

**Proof:** If \( K' \) is a synthesized key and \( B \wedge K' \), then by definition \( K \rightarrow A_f - K' \rightarrow B \in G_f^-(B) \). Thus, since \( B \wedge K, K' \rightarrow K \rightarrow G_f^-(B) \). It is given that \( K \rightarrow B \in G_f^-(B) \), and therefore \( K' \rightarrow B \in G_f^-(B) \).

**Superfluous attribute detection algorithm**

**Input.** \( R \), a preparatory relational schema; \( i \), the index of some scheme in \( R \); \( B \) an attribute in \( A_f \).

1. If \( K_f = \{A_f\} \)
   - then mark \( B \) non-superfluous and return
   - else mark \( B \) superfluous,

1.1. construct \( K_f' = \{K \wedge K_f | B \wedge K \} \)

1.2. construct \( G_f^-(B) \) by (temporarily) removing all dependencies involving \( B \) in \( R_i \) from \( G \)

2. (Check restorability.)
   - if \( K_f' \) is not empty
     - then choose any key \( K \) from \( K_f' \)

2.1. if \( K \rightarrow B \in G_f^-(B) \)
   - then mark \( B \) non-superfluous and return

3. (Check non-essentiality.)
   - For each key \( K \) in \( K_f - K_f' \) and while \( B \) is marked superfluous do
3.1. If $K \rightarrow A_i \notin G_i(B)^+$
3.1.1. then let $M$ denote the closure of $K$ relative to $G_i(B)$
3.1.2. if $(M \cap A_i) - B \rightarrow A_i \notin G_i^+$
then mark $B$ non-superfluous
3.1.3. else insert into $K'_i$ any key of $R_i$ contained in $(M \cap A_i) - B$
Output. $K'_i$ if $B$ is marked superfluous and $\emptyset$ if $B$ is marked non-superfluous.

Each substep for Step 3.1 runs in time $O(|G| + |A_i|)$ and is executed at most $O(|K_i|)$ times. Since Step 2.1 runs in time of the same order and is executed at most once, Steps 2 and 3 together take $O(|K_i| + |F| + |A_i|)$ time (at most one dependency in $G$ results from each given functional dependency). Because Step 1.1 takes time $O(|K_i|)$ and Step 1.2 takes time $O(|F|)$, a single attribute is checked in $O(|K_i| + |F| + |A_i|)$ time.

**Theorem 5.** Let $R$ be a preparatory relational schema including $R_i$, and let $B_i$ be an attribute in $A_i$. If $B$ is not superfluous in $R_i$, then it will not be superfluous in any relation scheme derived from $R_i$ by the removal of superfluous attributes from a scheme in $R$.

**Proof:** The details of this proof are too lengthy to include here. The following statements highlight the arguments:

1. For two attributes $B_1$ and $B_2$ in $R_i$, the part of $D_i(B_1)$ that does not involve $B_2$ in $R_i$ is identical to the part of $D_i(B_2)$ that does not involve $B_1$ in $R_i$. Furthermore, the closure of that part is contained in both $D_i(B_1)^+$ and $D_i(B_2)^+$.
2. Because of 1, an attribute that is not restorable in $R_i$ cannot become restorable through the removal of other attributes in $A_i$.
3. Let $B_1$ be an attribute that is essential in $R_i$ and $K'_i$ be the set of keys from which $A_i$ cannot be derived using $D_i(B_1)$. If another attribute $B_2$ is in all keys in $K'_i$, then $B_2$ is also essential.
4. From 1 and 3, an attribute that is essential in $R_i$ cannot become non-essential through the removal of other non-essential attributes in $A_i$.
5. Removal of attributes from other schemes in $R$ does not affect whether or not $B$ is superfluous in $R_i$.

The result of Theorem 5 is that each attribute in $R$ must be tested once only; once found to be non-superfluous it need not be re-examined after removing other attributes. Hence the complete normalization algorithm is as follows:

**Deletion normalization algorithm.**

**Input.** $A$, a set of attributes; $F$, a set of functional dependencies on $A$.

1. Prepare a relational schema.
2. Use the preparatory algorithm for $A$ and $F$ to yield $R$.
3. (Test each relation scheme for superfluous attributes.)
   For $i := 1$ to $|R|$ do
   3.1. (Test each attribute in $A_i$)
      For each $B$ in $A_i$ do
         If the superfluous attribute detection algorithm returns a non-essential then
         for $R_i$, $i$, and $B$ then
      3.1.1. Construct $R'_i$ such that $A'_i = A_i - B$ and $K'_i$ is the removed attributes.
2.1.2. Replace \( R_i \) by \( R'_i \) in \( R \).

Output. \( R \), a relational schema in improved third normal form.

Given a particular relation scheme \( R_i \), because in Step 2.1 the superfluous attribute detection algorithm is called for each attribute in \( A_i \), the time for that step is bounded by \( O(|K_i| |F| |A|^{3}) \). (In Step 2.1.1, the size of \( K_i \) is always less than or equal to the size of \( K_i \) since the algorithm introduces at most one new key for each key removed by the elimination of \( B \), as implied by the proof for Theorem 3.) Step 2.1, in turn, is repeated for each relation scheme in \( R \), where each functional dependency results in the appearance of at most one new key in some one scheme in \( R \). Since the number of keys in total is therefore bounded by \( |F|+1 \) (the extra key resulting from Step 4 of the preparatory algorithm where a relation scheme may be inserted into \( R \) for reconstructibility), Step 2 takes time \( O(|F|^{2}|A|^2) \). Because Step 1 also requires time of the same order, that is the bound for the complete algorithm.

5. Conclusions

We have shown that some Codd third normal form relation schemes and even some Boyce-Codd normal form schemes still contain simply removable superfluous attributes because the definitions of transitive dependency and non-prime attribute are inadequate when applied to sets of schemes. We defined restorable and non-essential to replace those definitions and proved that in a preparatory relational schema, a transitive dependency implies the presence of a restorable attribute and an attribute that is essential is always prime. We were then able to define an improved third normal form which is superior to Codd third normal form in removing superfluous attributes. Furthermore we have proven that no superfluous attributes remain. We then presented the deletion normalization algorithm which produces a relational schema in improved third normal form while guaranteeing covering and reconstructibility. The complexity of the algorithm is \( O(|F|^{2}|A|^{2}) \) which is the same as that of the best known algorithms for generating relation schemes in Codd third normal form.

It should be noted that the removal of all superfluous attributes does not necessarily imply the absence of update anomalies. In particular, if an attribute \( B \) in \( R_i \) satisfies the covering condition but not the reconstructibility condition, then although it is not superfluous, its updating may lead to anomalous behaviour.

It is interesting to contrast the deletion normalization method with the decomposition method for normalization [Codd 71, Delobel 73, Rissanen 77]. Decomposing a relation scheme \( R \) into two schemes \( R_1 \) and \( R_2 \) requires that \( R \) can be reconstructed from \( R_1 \) and \( R_2 \). Let \( A_1, A_2 \) and \( A \) be the sets of attributes for \( R_1, R_2 \), and \( R \), respectively, and let \( F \) be the set of functional dependencies for \( R \). The reconstructibility of \( R \) requires a lossless join of \( R_1 \) and \( R_2 \), which has been shown to occur if and only if either \( A_1 \) or \( A_2 \) is functionally dependent on their intersection, that is, the intersection contains a key of \( R_1 \) or of \( R_2 \) [Rissanen 77, Aho 79]. Without loss of generality, assume that \( A_1 \cap A_2 \rightarrow A_2 \in F^+ \). Thus \( A \rightarrow A_1 \cup A_2 \) or \( A_1 \rightarrow A \in F^+ \), which implies that each key of \( R_1 \) is also a key of \( R \). Since \( A_1 \cap A_2 \rightarrow A_2 \rightarrow A_1 \) (\( = \neq A \) \( \neq A \)) \( \in F^+ \), \( X = A \cap A_2 \) is a set of attributes in \( R_1 \) and \( R_2 \) such that all those attributes which are in \( R \) (and \( R_2 \)) but not in \( R_1 \) are functionally dependent on \( X \) in \( R \). If \( R_1 \) and \( R_2 \) are distinct relation schemes in a preparatory relational schema, then \( X \rightarrow K \in F^+ \) where \( K \) is a key of \( R_1 \) (otherwise the relations must be combined). When \( R \) is (non-trivially) decomposed into \( R_1 \) and \( R_2 \), there is at least one attribute in \( r \) in \( A \rightarrow A_1 \). In this case, \( K \rightarrow X \in F^+ \), \( X \rightarrow B \in F^+ \), and \( X \rightarrow K \in F^+ \), that is, \( K \rightarrow B \in F^+ \) is a transitive dependency in \( R \). Hence the decomposition method for normalizing a relation scheme is applicable if and only if there exists a transitive dependency within the relation scheme. Furthermore, if \( B \) is transitive dependent on some key in \( R \), the result of applying the deletion normalization algorithm to \( B \) is the same as the result of applying decomposition. Thus the deletion normalization method is more powerful than the decomposition method for normalization.

The definitions suggested here can easily be extended to give an improved version of Boyce-Codd normal form as follows:

A relation scheme \( R_i \) in a relational schema \( R \) is in [improved Boyce-Codd normal form if
It can easily be shown that any relation scheme in improved Boyce-Codd normal form is also in both Boyce-Codd normal form and improved third normal form. It must be remembered, however, that for a given $A$ and $F$, a given Boyce-Codd normal form may not exist, and thus it is not necessarily possible to find a relation scheme in relational scheme each element of which is in improved Boyce-Codd normal form.

Finally, for the purposes of this paper, we have assumed that a functional dependency has been restored to functional dependencies only for the sake of simplicity in presentation. At least in the constraints of the model, the constraints of the dependency theory and the constraints of the relational model. Further research is needed to determine the feasibility of the recovery of those constraints.

References


Beeri, C. and Bernstein, P.A. Computational problems related to the design of normal form relational schemes. *ACM Trans. on Database Systems* 4, 1 (March 1979) 30-59.


