Conditions of Efficient Updates on Network Structures

Shin’ichi KONOMI, Nonmember Tetsuya FURUKAWA and Yahiko KAMBAYASHI, Members

Abstract
Redundant network structures allow efficient retrievals. Procedures to design network structures that allow efficient retrievals\(^9\) add redundancies into network structures. However, as redundancies increase, costs of updates increase. When network structures are redundant, costs of updates include costs of updating original data, checking consistencies and updating derived data. In this paper, we present the super-key condition to reduce costs of checking consistencies in redundant network structures.

First, we present constraints required to keep redundant network structures consistent and show a sequence of processes to delete or insert a record in redundant network structures keeping the constraints. Then, Relationships between functional dependencies and updates are discussed to derive the super-key condition. The condition classifies network structures regarding costs of the processes to delete or insert a record. Finally, we give a procedure to design network structures that satisfy the super-key condition. The procedure may increase redundancies of network structures to reduce costs of updates.

1. Introduction
Redundancy of information is an important factor to discuss costs of retrievals and updates. Databases that are designed based on semantic constraints of data, such as functional dependencies, generally have less redundancies. It is easy to maintain consistencies of databases after updates if redundancies of the databases are small. Kuck and Sagiv\textsuperscript{(4)} discuss designs of network databases based on functional dependencies. On the other hand, redundancies can serve fast data retrievals. A simple example of this situation is to store results of retrievals as redundant data in databases. In network databases, costs of retrievals are reduced by adding redundant record types and DBTG sets\textsuperscript{(9)} in network schemas.

Link-structured databases can be customized to perform fixed-form processes efficiently. Relational databases have simple data structures and simple mathematical model. For efficiencies of processes, however, link structures are often introduced in relational databases in the physical level. Join operations are essential and expensive operations to perform retrievals in relational databases. Indices\textsuperscript{(2,8)} and hash functions\textsuperscript{(7,3)} are used to allow efficient join operations, which increases redundancies in databases. Network databases have network data structures that consist of records and links. Retrievals can be performed efficiently using links. Both indices and network structures have links that allow efficient retrievals. Hash functions that do not have physical links can also be considered to have links logically. Network structures of network databases are used as the model in the following part of the paper. However, the idea of the paper can be applied to data structures that have links, such as indices, hash functions of relational databases or data structures of references in object-orient databases.
Designers of network structures should be sensitive about typical retrievals and updates since efficiencies of retrievals and updates depend on designs of network structures. Network structures permitting efficient retrievals can be designed by adding redundant structures\(^{(9)}\). However, redundant structures increase costs of updates. Designs that decrease not only costs of retrievals but also costs of updating redundant structures are required. In this paper, we formalize processes required for updates on network structures, present the super-key condition that is the condition satisfied by network structures permitting efficient updates, and show a procedure to design network structures satisfying the condition.

Data in one part of a redundant network structure may also be retrieved in another part of the network structure. One of these parts are called an original part and the other a derived part. Since data in derived parts of network structures should be consistent with data in original parts, derived parts should reflect updates in original parts. There exist two schedules to update original parts and derived parts. If derived parts are updated first and then original parts are updated, there may not be enough information to update original parts, which is the same case as view update problem\(^{(1,5)}\). We assume that original parts are updated first and then derived parts are updated.

Suppose record \(r\) is inserted, deleted or modified in a network structure. First, the corresponding records in derived parts of the network structure are obtained. These records are derived from linkages of records including \(r\). Second, since \(r^+\), one of the obtained records may be derived from another linkages of records not including \(r\) and may not require an insertion or a deletion, each of \(r^+\)
needs to be examined by retrievals. These retrievals can be performed efficiently if derived parts of network structures contain a set of records that correspond to linkages of records in original parts one by one.

2. Network Structures and Retrievals

Network structures can be represented by record types and DBTG sets where record types are sets of attributes and DBTG sets are one-to-many relationships of records. An attribute is an element of a finite set $U=\{A_1,A_2,\ldots\}$. Each attribute $A_i$ has its domain $\text{DOM}(A_i)$. In the following part of the paper, $A$, $B$, $C$, ... and $X$, $Y$, $Z$, ... represent attributes and sets of attributes, respectively. Concatenation of these symbols represents union, e.g., $ABC$ represents set of attributes $\{A, B, C\}$. Equivalences of record types can be defined in two manners. $X=Y$ denotes attributes of $X$ and $Y$ are identical while $X\equiv Y$ denotes $X$ and $Y$ are the same record type. $X\equiv Y$ implies $X=Y$.

Definition 1
Let $X$ be a record type where $X=A_1A_2\ldots A_n$. A record $r(X)$ of record type $X$ is a mapping

$$r: (A_1,A_2,\ldots,A_n) \rightarrow \text{DOM}(A_1) \cup \text{DOM}(A_2) \cup \ldots \cup \text{DOM}(A_n)$$

where $r$ maps $A_i$ to $\text{DOM}(A_i)$.

$r(X)$ can be abbreviated to $r$ when $X$ is trivial. $F(X)$ represents a file that is a set of records of $X$. The domain of $F(X)$ is $\text{DOM}(A_1) \times \text{DOM}(A_2) \times \ldots \times \text{DOM}(A_n)$.

Definition 2
Let $S<X, Y>$ be a DBTG set where $X$ is the owner record type of member record type $Y$. An occurrence of the DBTG set $S<X,Y>$, denoted $i(X,Y)$, is an one-to-many relationship between record $r(X)$ and records $r_1(Y), r_2(Y), \ldots, r_n(Y) \ (n \geq 1)$.

$r$ is the owner record of member records $r_1, r_2, \ldots, r_n$. $I(X,Y)$ represents a set of occurrences of DBTG set $S<X,Y>$.

Definition 3
A network schema is a connected directed graph $B(V,E)$. $V$ is a set of vertices and $E$ is a set of directed edges, where vertices and edges correspond to record types and DBTG sets, respectively. The edges are directed from owner record types to their child record types.

A network structure is represented by a network schema $B$. An instance of network schema $B(V,E)$ is the files of the record types in $V$ and sets of occurrences of the DBTG sets in $E$. Figure 1 shows an example of a network schema and its instance. One-to-many relationships of records can be represented with lines from an owner record to its member records as in the figure. We do not discuss differences of implementations of DBTG sets.
Figure 1: Network schema and corresponding instance values.

Example 1

Figure 1 (a) is network schema \( B(V,E) \) where \( V=\{A,B,C,DE,F\} \) and \( E=\{S_1<A,B>,S_2<C,B>,S_3<C,DE>,S_4<F,DE>\} \). Vertices A, B, C, DE and F represent record types and the edges represent DBTG sets. Figure 1 (b) is an instance of the schema where values of attributes are represented by the corresponding lowercase letters. File F(A) consists of records \( a_1 \) and \( a_2 \), file F(DE) consists of records \( d_1e_1, d_2e_1, d_3e_3 \), etc. Lines between records represent occurrences of DBTG sets, relating one owner record to its member records.

Record types and files in network schemas correspond to relational schemas and relations in relational databases, respectively. Let \( X \) be set of attributes \( A_1A_2...A_n \) and \( R(X) \) be a relation on \( X \) where \( R(X) \) is a finite set of mappings \( t \) such that
\[
t:(A_1,A_2,...,A_n)\rightarrow\text{DOM}(A_1)\cup\text{DOM}(A_2)\cup...\cup\text{DOM}(A_n)\text{ and }t\text{ maps }A_i\text{ to }\text{DOM}(A_i).\]
\( X \) is a relational schema of a relation \( R \) whose domain \( \text{DOM}(X) \) is \( \text{DOM}(A_1)\times\text{DOM}(A_2)\times...\times\text{DOM}(A_n) \). \( t\in R(X) \) is called a tuple of relation \( R \) where \( t=a_1a_2...a_n \ (a_i\in\text{DOM}(A_i)) \).
Relations can be represented by tables whose rows and columns correspond to tuples and attributes, respectively. As operations applied to relations, selections select a tuple from a relation, projections project a relation on some attributes and joins merge relations. These operations are described by relational algebra.

Let $\theta$ be one of comparative operators, $=, >, <, \geq, \leq$ and $\neq$, $Y$ be set of attributes $B_1 B_2 ... B_m \subseteq X$, $b_j \in \text{DOM}(B_j)$ and $t[Y]$ be such a set of values $b_1 b_2 ... b_m$ that $b_j = a_i$ if $B_j = A_i$. $t[Y]$ is such a set of values that are obtained by discarding values of $X-Y$ form $t$.

Selection
Let $R(X)$ be a relation, $Y(\subseteq X)$ be attributes $B_1 B_2 ... B_m$, and a constant $'c' \in \text{DOM}(B_1) \times \text{DOM}(B_2) \times ... \times \text{DOM}(B_m)$. Selection of $R$ by condition $Y \theta 'c'$ is $R[Y \theta 'c'] = \{t \mid t \in R, t[Y] \theta 'c'\}$.

Projection
Let $R(X)$ be a relation and $Y(\subseteq X)$ be attributes. Projection of $R$ on $Y$ is $R[Y] = \{t[Y] \mid t \in R\}$.

(Natural) Join
Let $R_1(X_1)$ and $R_2(X_2)$ be relations. Natural join of $R_1$ and $R_2$, denoted $R_1 \ast R_2$, is relation $R(X_1 \cup X_2) = \{t \mid t[X_1] \in R_1, t[X_2] \in R_2\}$.

Let set of vertices of $B(V,E)$ be $\{X_1, X_2, ..., X_n\}$, $r_i$ be a record of record type $X_i$ and $t$ be $\{r_1, r_2, ..., r_n\}$. If $S<X_i,X_j>$ does not exist or $r_i$ is the owner record of $r_j$, $t$ is a tuple of records on $B$, denoted $t(B)$. Let $R(B)$ denote such a relation $R(X_1 \cup X_2 \cup ... \cup X_n)$ that each tuple corresponds to a tuple of records on $B$. If $V=\{X\}$ and $E=\phi$, $R(B) = F(X)$, denoted by $R(X)$. 
Let \( R(\mathbf{B}) \) be a relation on \( \mathbf{B} \) and \( Y, Z \subseteq \text{att}(\mathbf{B}) \). If \( t_1[Y] \neq t_2[Y] \) or \( t_1[Z] = t_2[Z] \) for \( \forall t_1, t_2 \in R(\mathbf{B}) \), \( Y \) functional determines \( Z \), denoted \( Y \rightarrow Z \). A value of \( Y \) corresponds only one value of \( Z \) if \( Y \rightarrow Z \). \( Y \rightarrow Z \) holds if \( Y \supseteq Z \). Also, \( X \rightarrow Z \) holds by transitivity of functional dependencies if \( X \rightarrow Y \) and \( Y \rightarrow Z \).

**Definition 5**

If \( Y \) is such a minimal subset of attributes \( X \) that \( Y \rightarrow X \), \( Y \) is a key of record type \( X \), denoted \( \text{key}(X) \). Also, if \( Y \) is such a minimal subset of \( \text{att}(\mathbf{B}) \) that \( Y \rightarrow \text{att}(\mathbf{B}) \), \( Y \) is a key of network schema \( \mathbf{B}(V,E) \), denoted \( \text{key}(\mathbf{B}) \).

DBTG sets of network schemas represent functional dependencies from member record types to their owner record types. The functional dependencies represented by DBTG sets and keys of record types determine keys of network schemas.

**Example 2**

In Figure 1 (a), underlined attributes are the key in each record type. \( \mathbf{B} \rightarrow \mathbf{AC} \) and \( \mathbf{D} \rightarrow \mathbf{CF} \) are functional dependencies represented by DBTG sets. \( \mathbf{D} \rightarrow \mathbf{E} \) since the key of record type \( \mathbf{DE} \) is \( \mathbf{D} \). \( \mathbf{BD} \rightarrow \mathbf{BD} \) is trivial. Then, since \( \mathbf{BD} \rightarrow \mathbf{ABCDEF} \), a key of the network schema in figure 1 (a) is \( \mathbf{BD} \).

**Definition 6**

A network subschema of \( \mathbf{B} \), denoted \( \mathbf{B}' \), is a network schema represented by a subgraph of \( \mathbf{B} \).
We assume that $B$ is a directed acyclic graph and $R(B)[\text{att}(B')] = R(B')$.

**Definition 7**

Let $B'(V', E')$ be a network subschema of $B(V, E)$. $X_S$ and $X_P$ be subsets of $\text{att}(B)$, and $R(B')$ be a relation on $B'$. Queries that are evaluated by selection $R[X_S='c']$ and projection $R[X_P]$ are retrieval queries denoted by $Q(X_S, X_P)$. $X_S$, $X_P$, and $X_Q(=X_S \cup X_P)$ are the selection attributes, projection attributes and the query attributes, respectively. $B'$ is a query network schema of $Q$ on $B$, denoted by $B_Q$, if $B'$ is such a minimal network subschema of $B$ that $Q$ can be evaluated only by selections and projections of $R(B')$.

Generally network structures can be customized for efficient retrievals by adding redundant record types and redundant DBTG sets. For example, record type $X_Q$ helps to evaluate retrieval query $Q$. Records of record type $X_Q$ are derived from an original part of a network structure. File $F(X_Q)$ is $R(B)[X_Q]$, i.e. projection of $R(B)$ on $X_Q$ and an owner record types of $X_Q$ is $X$ such that $X_Q \rightarrow X$. Query $Q$ can be retrieved only from $F(X_Q)$.

**Example 3**

Figure 2 shows a network structure designed to allow efficient retrieval of query $Q(A,F)$. Compared with Figure 1 (a), record type $AF$ is added. Corresponding values of attributes $A$ and $F$ are quickly retrieved in file $F(AF) = \{a_1f_1, a_1f_2, a_2f_2\}$. 
Redundant record types as $F(AF)$ increase redundancies of network structures and thus increase time to perform updates.

**Definition 8**
Let $B(V,E)$ and $B^+(V^+,E^+)$ be network schemas such that $\text{att}(B^+) \subseteq \text{att}(B)$ and $B^*(V^*,E^*)$ be a network schema such that $V^* = V \cup V^+$ and $E^* = E \cup E^+$. An instance of $B^*$ is consistent when constraints (i) holds.

$$R(B^+) = R(B)[\text{att}(B^+)] \quad (i)$$

**Example 4**
Let the network schema of Figure 1 (a) be $B(V,E)$ and the network schema of Figure 2 (a) be $B^*(V^*,E^*)$. Then, $V^+ = \{A,AF,F\}$, $E^+ = \{S_5<A,AF>, S_6<F,AF>\}$, $V^* = \{A,B,C,DE,F,AF\}$, $E^* = \{S_1<A,B>, S_2<C,B>, S_3<C,DE>, S_4<F,DE>, S_5<A,AF>, S_6<F,AF>\}$.

3. Updates in Redundant Network Structures

Updates in network structures can be classified as follows:
(1) Updates that do not change relationships of records, such as modifications of an attribute value.

(2) Updates that change relationships of records.
   (2-1) Insertions of a record or a link
   (2-2) Deletions of a record or a link
   (2-3) Modifications of a link (may be performed by deletions of a link and insertions of a link)

When network structures are redundant, constraint (i) should be forced to keep network structures consistent. Updates of $B$ require $B^+$ to reflect them. Reflection of updates requires retrievals to find and test data that may be updated in $B^+$. However, every update in $B$ does not necessarily affects $B^+$. $B^+$ has the original part in $B$ and the updates in $B$ must be the updates in the original part of $B^+$ when $B^+$ is affected.

Definition 9
The original part of network schema $B^+$ is a minimal network subschema of $B$ such that $\text{att}(B') \supseteq \text{att}(B^+)$. Also, the original part of attributes $X$, denoted $B'_X$, is a minimal network subschema of $B$ such that $\text{att}(B) \supseteq X$.

Lemma 1
Let $B'$ be the original part of $B^+$. If $\text{R}(B^+) = \text{R}(B')[\text{att}(B^+)]$, the constraints of (i) is satisfied, i.e., $\text{R}(B^+) = \text{R}(B)[\text{att}(B^+)]$.

Proof.
Since $B'$ is a network subschema of $B$

$$\text{R}(B') = \text{R}(B)[\text{att}(B')]$$
att(B') \supset att(B \setminus s(,+)) permits to project both sides of the equation to att(B^+).

\[ R(B')[\text{att}(B^+)] = R(B)[\text{att}(B')][\text{att}(B^+)] \]

Since \( R(X)[Y][Z] = R(X)[Z] \) if \( X \supseteq Y \supseteq Z \), this equation is equivalent to

\[ R(B')[\text{att}(B^+)] = R(B)[\text{att}(B^+)]. \]

Thus, \( R(B^+) = R(B)[\text{att}(B^+)] \) if \( R(B^+) = R(B')[\text{att}(B^+)]. \)

When record \( r \) of record type \( X \) in \( B' \) is inserted, deleted or modified, \( B^+ \) satisfies constraint (i) after applying the following steps for update propagation.

Update propagation

These three steps are executed for all the record types \( X^+ \) in \( B^+ \).

1. Candidate retrieval
   \( Q(X,X^+) \) is evaluated to retrieve such records of \( X^+ \) that are derived from records including \( r \). The answer of \( Q \) is a set of candidate records \( \{r_1, r_2, \ldots, r_n\} \).

2. Derivation test
   For each \( r_1(1 \leq i \leq n) \), \( Q(X^+,X) \) is evaluated to find such a set of records that derives \( r_1 \) and does not include \( r \).

3. Pure update
   \( r_1 \) is inserted or deleted if \( r_1 \) does not pass the derivation test, i.e., the set of records exists.
These steps can be represented by relational algebra with \( R(\mathbf{B'}) \), a relation of the original part of \( \mathbf{B}^+ \). Projection of selection \( R(\mathbf{B'})[X=r] \) corresponds to Step 1. Note that Step 1 is performed by navigation using records and links.

Because candidate record \( r_i(1 \leq i \leq n) \) obtained in Step 1 may still be derived from such tuples in \( R(\mathbf{B'}) \) that are irrelevant to \( r \), derivation test is required. Step 2 performs navigations that correspond to \( R(\mathbf{B'})[X_i=r_i][X] \) for each \( r_i(1 \leq i \leq n) \). Note that the navigations can terminate when the number of obtained records is more than one.

Figure 3 shows steps to update redundant network schema \( \mathbf{B}^* \). Derivation test occupies high percentages of costs of updating \( \mathbf{B}^* \) since it is executed for each records obtained in the candidate retrieval.

Example 5

Let us consider updates in the network structure of Figure 2. The network structure, represented by \( \mathbf{B}^* \), is a redundant network.
structure as detailed in Example 4. The original part of $B^+$, denoted by $B'$, is $B$ itself since $\text{att}(B^+)=\Lambda F$.

Suppose record $b_2$ of record type $B$ is deleted. Navigations starting from $b_2(s(,2))$ give tuples of records $\{a(s(,1))\backslash b(s(,2))\backslash c(s(,1))d_1e_1f_1a_1f_1\backslash \}, \{a_1b_2c_1d_2e_1f_2a_1f_2\}$ (candidate retrieval). Then, the candidate records are records $a_1f_1$ and $a_1f_2$. Navigations starting from $a_1f_1$ or $a_1f_2$ give tuples of records $\{a_1b_1c_1d_1e_1f_1\}, \{a_1b_1c_1d_2e_1f_2\}$ except for the tuples obtained in the candidate retrieval (derivation test). According to the result of derivation test, $a_1f_1$ and $a_1f_2$ should not be deleted.

4. Network Structures Allowing Efficient Updates

We can customize network structures for efficient candidate retrieval and derivation test, and thus allow efficient updates of redundant network structures. Retrievals in network structure, which candidate retrieval and derivation test require, are classified as follows.

1. **Navigation retrieval**
   
   Retrieval that requires to navigate links of DBTG sets.

2. **In-file retrieval**
   
   Retrieval that does not require to navigate links of DBTG sets.

Lemma 2

Let $X^+$ be a record type in $B^+$, $B_{X^+}$ be the original part of $X^+$ and $X$ be a record type in $B_{X^+}$. Suppose record $r$ of record type $X$ is inserted, deleted or modified. If $X$ is a record type such that $X^+ \rightarrow X$, $B^+$...
satisfies constraint (i) after applying only candidate retrieval and pure update.

Proof. When a value of $X^+$ is assigned, a value of $X$ is uniquely determined since $X^+ \rightarrow X$. Because the number of the values of $X$ that are the answers of $Q(X^+, X)$ is always one and derivation test is for checking if the number of values of $X$ returned by $Q$, derivation test can be omitted.

Lemma 3
Let $X^+$ be a record type of $B^+, B^+_X$ be the original part of $X^+$ and $X$ be a record type in $B^+_X$. Suppose record $r$ of record type $X$ is inserted, deleted or modified. If $X^+$ is a record type such that $X^+ \sqsupseteq \text{key}(B^+_X)$, $B^+$ satisfies constraint (i) after applying only candidate retrieval and pure update.

Proof. $X^+ \rightarrow \text{key}(B^+_X)$ since $X^+ \sqsupseteq \text{key}(B^+_X)$. The definition of $\text{key}(B^+_X)$ leads $\text{key}(B^+_X) \rightarrow \text{att}(B^+_X)$. For every record type $X$ in $B^+_X$, $\text{att}(B^+_X) \rightarrow X$ since $\text{att}(B^+_X) \supseteq X$. Then, $X^+ \rightarrow \text{key}(B^+_X)$, $\text{key}(B^+_X) \rightarrow \text{att}(B^+_X)$ and $\text{att}(B^+_X) \rightarrow X$ imply $X^+ \rightarrow X$. According to Lemma 2, derivation test can be omitted.

Derivation test is required when the conditions of Lemma 2 and Lemma 3 is not satisfied. However, retrievals required for derivation test are always in-file retrieval if network schemas satisfy the following condition.
Definition 11

Let \( X + \) be a record type in \( B + \) and \( B(X +) \) be the original part of \( X + \). \( B^* \) satisfies the super-key condition if there exists such a record type \( X_K \) in \( B^* \) that \( X_K \supseteq \text{key}(B(X +)) \) for all \( X + \in V^+ \).

Example 6

Figure 4 shows a network schema that satisfies the super-key condition. The network schema in the figure is a redundant network schema where \( V^+ = \{B, BD, DE\} \), \( \text{key}(B_B) = B \), \( \text{key}(B_{BD}) = BD \) and \( \text{key}(B_{DE}) = DE \). \( B^* \) subsumes record types B, BD and DE that are super sets of \( \text{key}(B_B) \), \( \text{key}(B_{BD}) \) and \( \text{key}(B_{DE}) \) respectively.

Lemma 4

Let \( B^*(V^*, E^*) \) be a network schema that satisfies the super-key condition, \( X \) be a record type in \( B \) and \( X^+ \) be the only record type in
\( \mathbf{B^+} \) such that \( X^+ \supseteq \text{key}(\mathbf{B^+_X}) \) and \( X^+ \notin V \). Suppose record \( r \) of record type \( X \) is inserted, deleted or modified. \( \mathbf{B^+} \) satisfies constraint (i) after applying only candidate retrieval and pure update.

**Proof.** Since \( X^+ \supseteq \text{key}(\mathbf{B^+_X}) \), derivation test can be omitted according to Lemma 3.

**Example 7**
Suppose record \( b_2 \) of record type \( B \) is deleted in Figure 4. The network schema in the figure satisfies the super-key condition since \( BD \supseteq \text{key}(\mathbf{B_{BD}}) \). Candidate retrieval gives tuples of records \( \{b_2, b_2d_1\} \) and \( \{b_2, b_2d_2\} \). Then, the candidate records \( b_2d_1 \) and \( b_2d_2 \) are deleted.

**Definition 12**
Let \( S<X,Y> \) be a DBTG set. If candidate retrieval and derivation test is performed by retrieval query \( Q \) whose query attributes are \( XY \), the update propagation is called direct update propagation.

**Lemma 5**
Derivation test in direct update propagation can be performed by in-file retrievals.

**Proof.** Let \( X \) be an owner record type of \( Y \). Since \( Y \rightarrow X \), derivation test \( Q(Y,X) \) can be omitted according to Lemma 2. Derivation test \( Q(Y,X) \) can be performed by only examining the occurrences of DBTG set \( S<X,Y> \), which can be an in-file retrieval since derivation test does
not require really to retrieve records but only require to count the number of records.

Let $X_K$ be a record type such that $X_K \supseteq \text{key}(X^+)$ for all $X^+ \in V^+$. Direct update propagation is allowed if there exists $X_K$ in $B^*$. Since $X_K \rightarrow X^+$ holds for all record type $X^+$ in $B^+$, every $X^+ (\neq X_K)$ can be an owner record type of $X_K$.

**Theorem 1**

Let $B^* (V^*, E^*)$ be a network schema that satisfies the super-key condition and $X$ be a record type in $B$. Suppose record $r$ of record type $X$ is inserted or deleted. $B^*$ satisfies constraint (i) after applying candidate retrieval, derivation test that is performed without navigation retrievals, and pure update.

*Proof.* Let $B'$ be the original part of $B^+$ and $X_K$ be a record type such that $X_K \supseteq \text{key}(B_{X^+})$ for all $X^+ (\in V^+)$. $X_K$ can be updated without derivation test as in Lemma 4. Since $X_K \rightarrow X^+$ for all record type $X^+ (\neq X_K)$ in $B^+$, every $X^+$ can be an owner record type of $X_K$. Direct update propagation from $X_K$ to $X^+$ is allowed if it guarantees constraint (i). Then, update of all the record types in $B^+$ to satisfy constraint (i) requires only in-file retrieval as derivation test.

Now we prove that direct update propagation from $X_K$ to $X^+$ guarantees constraint (i), i.e., $R(B)[\text{att}(B^+)] = R(B^+)$. Since $X_K$ includes $\text{key}(B_{X^+})$, specifying a record of $X_K$ determines a tuple of $R(B')[X^+]$ and a record of $X^+$ uniquely. Also, every tuple of $R(B')[X^+]$ has corresponding records of $X_K$. A record of $X^+$ and records of $X_K$
that correspond to the same tuple of $R(B')[X^+]$ can be connected by
links. After updating record type $X_K$, tuples of $R(B')[X^+]$ exist iff
corresponding records of $X_K$ exist. Thus, after direct update
propagation from $X\setminus s(+)\$ to $X\setminus s(,K)\$, $R(B')[X\setminus s(+)\]$ is
satisfied. Let $V^+$ be $\{X_1^+, X_2^+, ..., X_n^+\}$. 

$$R(B')[att(B^+)] = R(B')[X_1^+ \cup X_2^+ \cup ... \cup X_n^+]$$
$$= R(B')[X_1^+]*R(B')[X_2^+]*...*R(B')[X_n^+]$$

Since $R(B')[X_i^+] = R(X_i^+)$ (1 \leq i \leq n),

$$R(B^+) = R(X_1^+)*R(X_2^+)*...*R(X_n^+)$$
$$= R(B')[X_1^+]*R(B')[X_2^+]*...*R(B')[X_n^+]$$
$$= R(B')[att(B^+)]$$

According to Lemma 1, $B^+$ satisfies constraint (i) if
$R(B')[att(B^+)] = R(B^+)$ according to Lemma 1.

5. Designing Network Structures with Super-Key Files

Efficiencies of retrievals and updates in network structures
depend on the designs of them. To allow fast retrievals,
redundancies may be introduced into network structures. When
redundancies in network structures increase costs of updates, super-
key files can be added to network structures to reduce the costs.

Let $B^*$ be a network structure that does not satisfy the super-
key condition. The following procedure adds record types to $B^*$ and
returns $B^{**}$, a network schema that satisfies the super-key
condition.

Procedure 1
Let $\mathbf{B}(V,E)$ be a network schema designed based on dependencies of data such as functional dependencies and $\mathbf{B}^*$ a network schema that is designed by adding $\mathbf{B}^+(V^+,E^+)$ to $\mathbf{B}$ to allow fast processing of retrieval queries $Q_1, Q_2, \ldots, Q_k$.

1. Obtain $\mathbf{B}_{X_1^+}$ for every record type $X_1^+ (\in V^+, 1 \leq i \leq n)$.

2. Add $X_K (\supseteq \text{key}(\mathbf{B}_{X_1^+}), 1 \leq i \leq n)$ in $\mathbf{B}^+$.

3. Let record types in $\mathbf{B}_{X_1^+} (1 \leq i \leq n)$ that do not have member record types be owner record types of $X_K$.

4. Let every $X_1^+ (1 \leq i \leq n)$ be an owner record type of $X_K$.

5. Let $\mathbf{B}^{**}$ be the resulting network schema.

**Theorem 2**
Suppose $\mathbf{B}^*$ is a resulting network schema of Procedure 1 and record $r$ of record type $X$ is inserted or deleted in $\mathbf{B}^*$. $\mathbf{B}^*$ satisfies constraint (i) after applying candidate retrieval, derivation test that is performed without navigation retrievals, and pure update.

**Proof.** Since $X_K$ is the record type such that $X_K \supseteq \text{key}(\mathbf{B}_{X_1^+})$ for all record type $X^+ (\in V^+)$, $\mathbf{B}^*$ satisfies the super-key condition. Also, it is possible to perform direct update propagation since $S<X_1^+,X_K>$ is added for every $X_1^+ (\in V^+)$ in Step 4. According to Theorem 1, $\mathbf{B}^*$ satisfies constraint (i) after applying candidate retrieval, derivation test that is performed without navigation retrievals, and pure update.

**Theorem 3**
Let \( Q_1, Q_2, \ldots, Q_k \) be retrieval queries, \( B_{Q_i} \) for \( 1 \leq i \leq k \) be a query network schema of \( Q_1, Q_2, \ldots, Q_k \) in \( B_{**} \) of Procedure 1 and \( B_{Q_i} \) for \( 1 \leq i \leq k \) be a query network schema of \( Q_1, Q_2, \ldots, Q_k \) in \( B_{*} \) of Procedure 1. \( B_{Q_i} \) can be a network subschema of \( B_{Q_i} \).

**Proof.** \( B_{Q_i} \) is a network subschema of \( B_{**} \) since all Procedure 1 does is to add record types and DBTG sets. \( Q_i \) can be retrieved in such network subschema \( B_{Q_i} \) that is equal to or subsumed by \( B_{Q_i} \).

Procedure 1 does not increase costs of retrieving \( Q\backslash s(1) \), \( Q\backslash s(2) \), \ldots, \( Q_k \) according to Theorem 3.

Example 5

The result of applying Procedure 1 to the network schema in Figure 2 (a) is in Figure 5. Record type BD is added to \( B\backslash s(1) \) since \( BD \supseteq \text{key}(B_{AF}) \). Also, \( S_{AF,BD} \) is added for direct update propagation from BD to AF. In Figure 5, retrieval query \( Q(A,F) \) can be evaluated in record type AF as in Figure 2.
6. Conclusion

Redundant network structures can be consistent after updates by candidate retrieval, derivation test and pure update of the candidate records. Derivation test does not require costly navigation retrievals when network schemas satisfy the super-key condition. Network structures can be designed to satisfy the super-key condition by Procedure 1.

Generally, redundancies of data decrease costs of retrievals and increase costs of updates. When network structures are redundant, updates require update propagation to keep data in redundant network structures consistent. In this case, however, redundancies of data can decrease costs of updates.

Even when redundancies can decrease costs of updates, costs of updates do not decrease forever as the increase of redundancies since the number of data to be updated increase. Optimal degree of redundancies are determined considering size of storages required for network structures as well as efficiencies of retrievals and
updates. Also, costs evaluation of designs satisfying super-key condition is required.

References


