International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-5, Issue-11)

Research Article **November 2016**

Some Properties of the Balance Schemes and Key with the Translation of Block Schemes

Trinh Dinh Thang
Faculty of Information Technology, Hanoi Pedagogical University N^{0}2,
Vietnam

Abstract-

The report proposed the concept of balance schemes in the database model of block form, prove properties of the balance schemes, algorithm for transforming the block schemes to the balance schemes, the relationship between balance schemes and key with the translation of block schemes... In addition, some properties related to the minimum left side, key and the translation of block schemes was also expressed and demonstrated here.

Keywords- key, block, block schemes, balance schemes, translation of block schemes.

I. THE DATABASE MODEL OF BLOCK FORM

A. The block, slice of the block

Definition I.1 [1]

Let R = (id; $A_1, A_2,..., A_n$) is a finite set of elements, where id is non-empty finite index set, A_i (i=1.. n) is the attribute. Each attribute A_i (i=1.. n) there is a corresponding value domain dom (A_i). A block r on R, denoted r(R) consists of a finite number of elements that each element is a family of mappings from the index set id to the value domain of the attributes A_i (i = 1.. n).

$t \in r(R) \Leftrightarrow t = \{ t^i : id \rightarrow dom(A_i)\}_{i=1..n}$.

The block is denoted r(R) or r(id; $A_1, A_2,..., A_n$), sometimes without fear of confusion we simply denoted r.

Definition I.2 [1]

Let R = (id; $A_1, A_2,..., A_n$), r(R) is a block over R. For each $x \in id$ we denoted $r(R_x)$ is a block with $R_x = (\{x\}; A_1, A_2,..., A_n)$ such that:

$t_x \in r(R_x) \Leftrightarrow t_x = \{t^i_x = t^i\big|_x\}_{i=1..n}$, $t \in r(R), t = \{t^i : id \rightarrow dom(A_i)\}_{i=1..n}$,

where $t^i_x(x) = t^i(x)$, i =1.. n.

Then $r(R_x)$ is called a slice of the block r(R) at point x.

B. Functional dependencies

Here, for simplicity we use the notation:

$x^{(i)} = (x; A_i)$; $id^{(i)} = \{x^{(i)} \mid x \in id\}$.

$x^{(i)}$ ($x \in id$, i = 1.. n) is called an index attribute of block scheme R = (id; $A_1, A_2,..., A_n$).

Definition I.3 [1]

Let R = (id; $A_1, A_2,..., A_n$), r(R) is a block over R, $X \rightarrow Y$ is a notation of functional dependency. A block r satisfies $X \rightarrow Y$ if for any $t_1, t_2 \in r$ such that $t_1(X) = t_2(X)$ then $t_1(Y) = t_2(Y)$.

Definition I.4 [1]

Let block scheme $\alpha=(R,F)$, R= (id; $A_1, A_2,..., A_n$), F is the set of functional dependencies over R. Then, the closure of F denoted F^+ is defined as follows:

$F^+ = \{ X \rightarrow Y \mid F \Rightarrow X \rightarrow Y \}$.

If $X = \{x^{(m)}\} \subseteq id^{(m)}$, $Y = \{y^{(k)}\} \subseteq id^{(k)}$ then we denoted functional dependency $X \rightarrow Y$ is simply $x^{(m)} \rightarrow y^{(k)}$.

The block satisfies $x^{(m)} \rightarrow y^{(k)}$ if for any $t_1, t_2 \in r$ such that $t_1(x^{(m)}) = t_2(x^{(m)})$ then $t_1(y^{(k)}) = t_2(y^{(k)})$,

where: $t_1(x^{(m)}) = t_1(x; A_m)$, $t_2(x^{(m)}) = t_2(x; A_m)$,
$t_1(y^{(k)}) = t_1(y; A_k)$, $t_2(y^{(k)}) = t_2(y; A_k)$.

Suppose we have f: $X \rightarrow Y \in F$ is functional dependency, then we denote:

$$LS(f) = X, RS(f) = Y,$$

$$\forall f \in F: \quad LS(F) = \bigcup_{f \in F} LS(f), \; RS(F) = \bigcup_{f \in F} RS(f).$$

Let R= (id; $A_1, A_2,..., A_n$), we denoted the subsets of functional dependencies over R:

$F_h = \{ X \rightarrow Y \mid X = \bigcup_{i \in A} x^{(i)}, Y = \bigcup_{j \in B} x^{(j)}$, $A, B \subseteq \{1,2,...,n\}$ và $x \in id\}$,

$F_{hx} = F_h \Big|_{\bigcup_{i=1}^{n} x^{(i)}} = \{X \rightarrow Y \in F_h \mid X, Y \subseteq \bigcup_{i=1}^{n} x^{(i)} \}$.

C. Closure of the index attributes sets:

Definition I.5 [2]

Let block scheme α= (R,F), R= (id; A_1, A_2,..., A_n), F is a set of functional dependencies over R.

For each $X \subseteq \bigcup_{i=1}^{n} id^{(i)}$, we define the closure of X for F denoted X^+ as follows:

$$X^+ = \{x^{(i)}, x \in id, i = 1..n \mid X \to x^{(i)} \in F^+\}.$$

D. Key of the block scheme α = (R, F).

Definition I.6 [2]

Let block scheme α=(R, F), R = (id; A_1, A_2,..., A_n), F is a set of functional dependencies over R, $K \subseteq \bigcup_{i=1}^{n} id^{(i)}$.

K called a key of block schema α if it satisfies two conditions:

a) $K \to x^{(i)} \in F^+$, $\forall x \in id$, i = 1..n.

b) $\forall K' \subset K$ then K' has no properties a).

If K is a key and $K \subseteq K''$ then K'' called a super key of the block scheme R for F.

E. The translation of the block scheme

Definition I.7 [3]

Let block schemes α= (R,F), β = (S,G), $X \subseteq \bigcup_{i=1}^{n} id^{(i)}$, $X = \{x^{(i)}, x \in id, i \in A\}$, $A \subseteq \{1,2, ..., n\}$. *We say that: the scheme β received from the scheme α through the translation according index attributes subset X, if after removing the attributes in X of scheme α then we obtain scheme β.*

To denote the translation from the scheme α to the scheme β according index attributes subset X, we write: $\beta = \alpha \setminus X$.

Actions remove X from the schema α to the schema β as follows:

1. *Determined $S = R \setminus X$, R = (id; A_1, A_2,..., A_n),* here we remove the attribute A_i ($i \in A$) in R, *the complexity of this procedure is O(nk), with k is number of elements in A.*
2. *For each functional dependency $M \to N$ in F, with M, $N \subseteq \bigcup_{i=1}^{n} id^{(i)}$*, it creates a new *functional dependency* $M\setminus X \to N\setminus X$ in G. This procedure is denoted $G = F \setminus X$ and its complexity is O(mnk) with m is number of functional dependencies in F.

From there we see the complexity of the translation $\beta = \alpha \setminus X = (R\setminus X, F\setminus X)$ is O(mnk). Therefore, it is linear according the length of the input data.

After performing the procedure $G = F\setminus X$ then:

+ If G contains trivial functional dependencies (form $X \to Y$, $X \supseteq Y$) then it excludes this dependencies in G.

+ If G contains same functional dependencies then it excludes duplicate of this dependencies (G does not contain the same dependencies).

We have the following comments:

Comment I.1 [3]**:**

Let block schemes α = (R,F), β = (S,G), $X \subseteq \bigcup_{i=1}^{n} id^{(i)}$, $X = \{x^{(i)}, x \in id, i \in A\}$, $A \subseteq \{1,2, ..., n\}$. *Scheme β is received from scheme α through the translation according attribute subset X: $\beta = \alpha \setminus X$.*

In case, if id ={x} then block scheme α degenerate into relational schemas and the translation according attribute subset X again become the translation according attribute subset X from relational scheme α to relational scheme β in the relational database model.

Comment I.2 [3]**:**

Let block schemes α = (R, F_h), β = (S, G_h), $X \subseteq \bigcup_{i=1}^{n} id^{(i)}$, $X = \{x^{(i)}, x \in id, i \in A\}$, $A \subseteq \{1,2, ..., n\}$. *If scheme β is received from scheme α through the translation according attribute subset X: $\beta = \alpha \setminus X$ then: $S = R \setminus X$, $G_h = F_h \setminus X = \bigcup_{x \in id} F_{hx} \setminus X$. Thence inferred: $G_{hx} = F_{hx} \setminus (X \cap \bigcup_{i=1} x^{(i)})$, $\forall x \in id$.*

So, in this case the translation of block scheme again become the translation of slice schemes.

II. RESEARCH RESULTS

A. Balance block schemes

Definition II.1:

Block scheme α = (R,F), R = (id; A_1, A_2, ... , A_n) called balance if functional dependencies set F satisfies the following two properties:

1) *Union of all the left side of the functional dependencies in F equal union of all its the right side and equal set $\bigcup_{i=1}^{n} id^{(i)}$.*
2) *F has natural shortened form.*

As we know, *F has natural shortened form means F satisfies the following conditions:*

- F contains no trivial dependencies, ie the dependencies of the form:

$$X \to Y \in F \text{ and } X \supseteq Y.$$

- Left and right sides of the functional dependencies in F does not intersect:

$\forall f \in F$: $LS(f) \cap RS(f) = \varnothing$.

- The left side of the functional dependencies in F different double, mean:

$\forall f, g \in F$: $LS(f) = LS(g) \Leftrightarrow f = g$.

Comment II.1:

Let block scheme $\alpha = (R,F)$, $R = (id; A_1, A_2, \ldots, A_n)$ is the balance block scheme.

In case, if id ={x} then the balance block scheme α degenerate into balance relational schema in the relational data model.

Proposition II.1:

Let block scheme $\alpha = (R, F_h)$, $R = (id; A_1, A_2, \ldots, A_n)$ is the balance block scheme. Then $\forall x \in id$, slice scheme at point x: $\alpha_x = (R_x, F_{hx})$ is also balance scheme.

Prove

Indeed, because F_h is the balance scheme so by definition we have: F_h *has natural shortened form* and $LS(F_h) = RS(F_h) = \bigcup_{i=1}^{n} id^{(i)}$. Thence inferred:

$$LS(F_h)\Big|_x = RS(F_h)\Big|_x = \bigcup_{i=1}^{n} id^{(i)}\Big|_x \quad \Rightarrow \quad LS(F_{hx}) = RS(F_{hx}) = \bigcup_{i=1}^{n} x^{(i)}. \qquad (1)$$

On the other hand, for F_h has natural shortened form=> F_{hx} has also natural shortened form. (2)

From (1) and (2) we have: $\alpha_x = (R_x, F_{hx})$ is the balance scheme $\forall x \in id$.

Proposition II.2:

Let block scheme $\alpha = (R, F_h)$, $R = (id; A_1, A_2, \ldots, A_n)$. If $\forall x \in id$, the slice scheme at point x: $\alpha_x = (R_x, F_{hx})$ is the balance scheme then $\alpha = (R, F_h)$ is also the balance scheme.

Prove

Indeed, from the assumption $\forall x \in id$, $\alpha_x = (R_x, F_{hx})$ is the balance scheme so by definition we have: F_{hx} has natural shortened form and $LS(F_{hx}) = RS(F_{hx}) = \bigcup_{i=1}^{n} x^{(i)}$.

Thence inferred: $\bigcup_{x \in id} LS(F_{hx}) = \bigcup_{x \in id} RS(F_{hx}) = \bigcup_{x \in id} (\bigcup_{i=1}^{n} x^{(i)}) = \bigcup_{i=1}^{n} id^{(i)}$. (3)

On the other side, because F_{hx} has natural shortened form should:

$F_h = \bigcup_{x \in id} F_{hx}$ has also natural shortened form. (4)

From the results (3) and (4) we have: $\alpha = (R, F_h)$ is the balance scheme.

From the propositions 2.1 and 2.2 we infer following necessary and sufficient conditions for balancing scheme:

Proposition II.3:

Let block scheme $\alpha = (R,F_h)$, $R = (id; A_1, A_2, \ldots, A_n)$. Then:

$\alpha = (R,F_h)$ is the balance scheme if and only if $\forall x \in id$, the slice scheme at point x: $\alpha_x = (R_x, F_{hx})$) is the balance scheme.

Proposition II.4:

Let block scheme $\alpha = (R,F_h)$, $R = (id; A_1, A_2, \ldots, A_n)$. Then:

a) *If n =1 then $\forall x \in id$, $\alpha_x = (R_x,F_{hx})$ and $\alpha = (R,F_h)$ are unbalanced schemes.*
b) *If $\alpha = (R,F_h)$ is the balance scheme then intersection of the keys in the scheme $\alpha_x = (R_x,F_{hx})$, $\forall x \in id$ is $U_{I\,x} = \varnothing$.*
c) *If $\alpha = (R,F_h)$ is the balance scheme then intersection of the keys in it is $U_I = \varnothing$.*

Prove

a) With n = 1 then $\forall x \in id$, $\alpha_x = (R_x,F_{hx})$ and $R_x = (x, A_1) = x^{(1)}$, mean the scheme $\alpha_x = (R_x,F_{hx})$ have only one index attribute. Thus, for F_{hx} have only 4 ability to create the following dependencies:

1. $x^{(1)} \to x^{(1)}$ (trivial)
2. $x^{(1)} \to \varnothing$ (trivial)
3. $\varnothing \to \varnothing$ (trivial)
4. $\varnothing \to x^{(1)}$

Therefore, we can only choose $F_{hx} = \varnothing$ or $F_{hx} = \{\varnothing \to A\}$.

The first case: $R_x = x^{(1)}$, $F_{hx} = \varnothing$ we have: $LS(F) = RS(F) = \varnothing \neq R_x$.

The second case: $R_x = x^{(1)}$, $F_{hx} = \{\varnothing \to A\}$ we have: $LS(F) = \varnothing \neq x^{(1)} = RS(F)$.

So in both cases, the slice schemes at the point $x \in id$: $\alpha_x = (R_x,F_{hx})$ are not balancing scheme.

b) If $\alpha = (R,F_h)$ is the balance scheme then according to the results of proposition 2.3: the slice schemes $\alpha_x = (R_x,F_{hx})$, ($\forall x \in id$) are the balance schemes. From that, we have:

$\forall f \in F_{hx}$: $RS(f) \cap LS(f) = \varnothing$

Inferred: $\forall f \in F_{hx}$: $RS(f) \setminus LS(f) = RS(f)$.

From that:

$M = \bigcup_{f \in F_{hx}} (RS(f) \setminus LS(f)) = \bigcup_{f \in F_{hx}} RS(f) = RS(F_{hx}) = R_x$.

According to the formula for intersection of the keys, we have:

$U_{Ix} = R_x \setminus M = R_x \setminus R_x = \varnothing$, $\forall x \in id$.

c) If $\alpha = (R,F_h)$ is the balance scheme then according to its properties, we have:

$\forall f \in F_h$: $RS(f) \cap LS(f) = \varnothing$

Inferred: $\forall f \in F_h$: $RS(f) \setminus LS(f) = RS(f)$. From that:

$M = \bigcup_{f \in F_h} (RS(f) \setminus LS(f)) = \bigcup_{f \in F_h} RS(f) = RS(F_h) = R$.

From formula for intersection of the key, we have:

$U_I = R \setminus M = R \setminus R = \varnothing$.

Consequence II.1:

Let block scheme $\alpha = (R,F_h)$, $R = (id; A_1, A_2, ... , A_n)$. In the case id = {x} then the block scheme α degenerate into the relational scheme in the relational data model and:

a) *If n =1 then $\alpha = (R,F_h)$ is the unbalanced relational scheme.*

b) *If $\alpha = (R,F_h)$ is the balance relational scheme then intersection of the keys in it is $U_I = \varnothing$.*

Let block scheme $\alpha=(R,F_h)$, $R = (id; A_1, A_2, ... , A_n)$, F_h is the full set of functional dependencies. Then, we have the algorithm for translation of block scheme α to following balance format:

Algorithm Balance

Format Balance(α)

Input: $\alpha = (R, F_h)$, F_h is the full set of functional dependencies.

Output: Balance scheme $\beta = (V,G_h)$.

Method

1. Calculate G_h := Natural_Reduce(F_h);
2. Choose x any $\in$ id, calculate the intersection of keys in $\alpha_x = (R_x,G_{hx})$:
   $U_{Ix} = \bigcup_{i=1}^{n} x^{(i)} \setminus RS(G_{hx})$;
3. Determined M;
   3.1 P:= $RS(G_{hx}) \setminus LS(G_{hx})$;
   3.2 $M = (PU_{Ix})^+_{Ghx}$;
   3.3 Create scheme $\beta_x = (V_x, G_{hx})$, $V_x = \bigcup_{i=1}^{n} x^{(i)}$.
4. While $M \neq \varnothing$ do
   4.1 Translation $\beta_x = (V_x,G_{hx})$ by M: $\beta_x = \beta_x \setminus M$;
   // $V_x := V_x \setminus M$, $G_{hx} := G_{hx} \setminus M$
   4.2 Excluded from G_{hx} the dependencies of the form: $X \rightarrow \varnothing$.
   4.3 Created group of the dependencies with the same left side in G_{hx} .
   From $X \rightarrow Y_1$, $X \rightarrow Y_2$,..., $X \rightarrow Y_k$ to $X \rightarrow Y_1Y_2...Y_k$;
   4.4 $M := V_x \setminus LS(G_{hx})$;
   endwhile;
5. Determined $\beta = \bigcup_{x \in id} \beta_x$, $\beta = (V,G_h)$.
   // $V := \bigcup_{x \in id} V_x$, $G_h := \bigcup_{x \in id} G_{hx}$
6. Return(β);
   End Balance.
   Here Natural_Reduce(F_h) is the natural shortened algorithm for functional dependencies subset F_h.

Proposition II.5:

Block scheme $\beta=(V,G_h)$ obtained after the implementation of the algorithm Balance is the balance scheme.

Prove

To prove $\beta = (V,G_h)$ is the balance scheme, we prove $\forall x \in id$ then $\beta_x = (V_x ,G_{hx})$ is the balance scheme, then as a result of proposition 2.3 we infer $\beta = (V,G_h)$ is the balance scheme.

We see that, after the first step of the algorithm is done, G_h was in the natural shortened form. From that inferred $\forall x \in id$: G_{hx} has natural shortened form.

On the other hand, we have:

P:=$RS(G_{hx}) \setminus LS(G_{hx})$; $M = (PU_{Ix})^+_{Ghx}$, G_{hx} has natural shortened form, so:

$\forall f \in G_{hx}$: $RS(f) \cap LS(f) = \varnothing$ => $RS(f) \setminus LS(f) = RS(f)$.

From that:

$U_{Ix} = V_x \setminus \bigcup_{f \in G_{hx}} (RS(f) \setminus LS(f))= V_x \setminus RS(G_{hx})$.

We have: $LS(G_{hx}) = \bigcup_{f \in G_{hx}} (LS(f) \setminus M) = \bigcup_{f \in G_{hx}} LS(f) \setminus M = LS(G_{hx}) \setminus M$,

Similar: $RS(G_{hx}) = RS(G_{hx}) \setminus M$.

We prove the equality: $LS(G_{hx}) \setminus M = RS(G_{hx}) \setminus M = V_x \setminus M$ according to the following diagram:

$LS(G_{hx}) \setminus M \subseteq V_x \setminus M \subseteq RS(G_{hx}) \setminus M \subseteq LS(G_{hx}) \setminus M$.

Indeed, in the first iteration we have:

a) $LS(G_{hx}) \setminus M \subseteq V_x \setminus M$, patently.

b) $V_x \setminus M \subseteq RS(G_{hx}) \setminus M$: if $A \in V_x \setminus M$ then $A \notin M = (PU_{Ix})^+_{Ghx} \supseteq U_{Ix}$, from that: $A \notin U_I = V_x \setminus RS(G_{hx}) \Rightarrow A \in RS(G_{hx})$ and $A \in RS(G_{hx}) \setminus M$.

c) $RS(G_{hx}) \setminus M \subseteq LS(G_{hx}) \setminus M$: if $A \in RS(G_{hx}) \setminus M$ then $A \in RS(G_{hx})$ and $A \notin M = (PU_{Ix})^+_{Ghx} \supseteq P$. Inferred: $A \notin P = RS(G_{hx}) \setminus LS(G_{hx}) \Rightarrow A \in LS(G_{hx})$, and so: $A \in LS(G_{hx}) \setminus M$.

From the second iteration on, in 4.2 steps of the algorithm, after removing the functional dependencies of form: $X \rightarrow \emptyset$ from G_{hx} then V_x and $RS(G_{hx})$ unchanged. From that, equality $V_x = RS(G_{hx})$ is still true, although $LS(G_{hx})$ can be reduced.

Thus, after step 4.2 then the property for balance can be violated. Loop while has balancing task for the three sets V_x, $RS(G_{hx})$ and $LS(G_{hx})$. Therefore, we must calculate the amount of the difference: $M := V_x \setminus LS(G\ )$ in steps 4.4. According to the lemma about non-primitive attributes we have $M \subseteq U_0 \Rightarrow$ set of the key $Key(\beta_x)$ unchanged during the translation.

If $M \neq \emptyset$ then we continue to translation scheme β_x according to the amount of difference M. Because first scheme is finite and the translations are reducing the size of the file V_x, $LS(G_{hx})$, $RS(G_{hx})$ so M will gradually to $\emptyset$ and the loop ends.

When the loop ends, we prove that the received scheme is balancing scheme.

Indeed, from $M = \emptyset \Rightarrow V_x = LS(G_{hx})$, on the other hand combined with the loop invariant:

$V_x = RS(G_{hx})$, we have: $LS(G_{hx}) = RS(G_{hx}) = V_x \Rightarrow \beta_x = (V_x ,G_{hx})$ is the balance scheme.

Proposition II.6:

Every block scheme $\alpha = (R, F_h)$ with F_h is the full set of functional dependencies, are moved to the form of balance $\beta = (V,G_h)$ satisfying properties:

$$Key(\alpha) = U_I \oplus Key(\beta)$$

where U_I is the intersection of keys in α. The translation algorithm has polynomial complexity according to the length of the input data $O(n^2m)$, where n is the number of attributes, m is the number of functional dependencies in F_{hx}, $x \in id$.

Prove

The algorithm Balance transfer block scheme $\alpha = (R, F_h)$ to the balance scheme $\beta =(V,G_h)$. In addition, based on the results of the proposition on the translation of block scheme according to the non-key attributes set and intersection of the keys [3], we have:

$$Key(\alpha) = U_I \oplus Key(\beta).$$

With every step of the algorithm Balance, its complexity is not too O(mn), M is the attributes set and the number of its elements does not exceed n; so if desired M become $\emptyset$ then the loop while need to make maximum the n times.

From that, the complexity of the algorithm is $O(n^2m)$.

Consequence II.2:

Let block scheme $\alpha=(R,F_h)$, $R=(id; A_1, A_2, \ldots , A_n)$. In the case id={x} then the block scheme α degenerate into the relational scheme in the relational data model and:

Every relational scheme $\alpha=(R,F_h)$ are moved to the form of balance $\beta=(V,G_h)$ satisfying properties: $Key(\alpha) = U_I \oplus Key(\beta)$, where U_I is the intersection of keys in α.

The translation algorithm has polynomial complexity according to the length of the input data $O(n^2m)$, where n is the number of attributes, m is the number of functional dependencies in F_h.

III. CONCLUSIONS

The results of the balancing scheme and keys with the translation of block scheme in the database model of block form has clarified the structure of the design in the this model. In the case of block reduces to relations then some results coincided with the results of many authors have been given for relations in the relational data model. Some other results are reviewed in separate cases Some other results are reviewed in separate cases of the functional dependencies set F in the block scheme as the subset of dependencies F_h, F_h full... On the basis of these results we can further study the relationship between types of different logic dependency with the translation of block scheme..., contributed to a more complete design theory of the database model of block form.

REFERENCES

[1] Nguyen Xuan Huy, Trinh Dinh Thang, "The database model of block forms", *Journal of Computer Science and Cybernetics*, T.14, S.3, 1998, 52-60.

[2] Trinh Dinh Thang, Tran Minh Tuyen, "The translation of block scheme and the present problem of the closure, key in the database model of block forms", *Proceedings of the National Conference XIII "Some the selected issues of Information Technology and Communication"*, Hungyen, Aug. 19-20, 2010, 276-286.

[3] Trinh Dinh Thang, Tran Minh Tuyen, "Key and key attributes set, non-key attributes set with translation of block schemes", *Proceedings of the National Conference on the 14th, "Some the selected issues of Information Technology and Communication* ", Cantho, Oct. 07-08, 2011, 159-170.