A Horizontal Partition of the Object-Oriented Database for Efficient Clustering

Chin-Wan Chung, Chang-Ryong Kim, and Ju-Hong Lee

Abstract

The partitioning of related objects should be performed before clustering for an efficient access in object-oriented databases. In this paper, a horizontal partition of related objects in object-oriented databases is presented.

All subclass nodes in a class inheritance hierarchy of a schema graph are shrunk to a class node in the graph that is called condensed schema graph because the aggregation hierarchy has more influence on the partition than the class inheritance hierarchy. A set function and an accessibility function are defined to find a maximal subset of related objects among the set of objects in a class. A set function maps a subset of the domain class objects to a subset of the range class objects. An accessibility function maps a subset of the objects of a class into a subset of the objects of the same class through a composition of set functions.

The algorithm derived in this paper is to find the related objects of a condensed schema graph using accessibility functions and set functions. The existence of a maximal subset of the related objects in a class is proved to show the validity of the partition algorithm using the accessibility function.

I. Introduction

During the last few years, object-oriented databases (OODBs) gained a considerable attention mainly because they reduce the semantic gap between real world concepts and data representation models. A major drawback of OODBs is the low speed of query execution, due to the sequential processing of independent entities.

In databases, the clustering of data is needed to store and retrieve local data efficiently. Generally, the clustering of data can be divided into two phases. The first phase is the partitioning of related data. The second phase is to rearrange data in the partition block so that data which are more likely accessed together are located closely to increase the performance.

There are several issues on the clustering of object-oriented databases such as the object arrangement in a cluster, the effect of an object access model upon a cluster, the dynamic reclustering, a simulation model for performance measurement, a physical data store model for a cluster, and access frequency measurement model. Several papers published on the clustering of the object-oriented databases discussed these issues [1, 3, 4, 12]. However, they assumed that there were already the partition blocks of related objects. For example, in ORION system[8], instances of the same class are clustered in the same physical segment. Furthermore, instances which belong to user-specified collection of classes are stored in the same physical segment. Consequently, it is important to derive a partition block prior to clustering.

The information for partitioning databases can be given by users. However, the information from users is not always correct and sufficient. Therefore, an automatic partition of databases is desirable. In this paper, we present a method to find the partition whose block includes objects related by the aggregation hierarchy[2](same as the part-of relationship or the class composition hierarchy).

The data model has an effect upon the partition of data. We discuss the partition method for each data model. After that, we derive a partition method for the object-oriented data model.

The relational database can be partitioned more freely compared with other types of databases because it does not have physical connections between data. The data of the relational database are related by value itself. There are two general methods of partitioning in the relational

Manuscript received July. 29,–1995; accepted November 8, 1995.

The authors are with Department of Information and Communication Engineering, Korea Advanced Institute of Science and Technology.

data model, the horizontal method and the vertical method. The horizontal method partitions a relation along its tuples. The vertical method partitions the attributes of the relation [10, 11, 13].

In the network model and the hierarchical model, there are also two methods of partitioning, the horizontal method and the vertical method. The vertical partition method chooses all record occurrences in each record type as a unit of partition. This is analogous to taking a relation as a unit in the relational model. Given a data structure diagram, such partition is equivalent to the isolation of record types by cutting access paths vertically with respect to the plane of the diagram. The horizontal partition method takes all the occurrences connected to each other through access paths between different record types as a unit of partition. Therefore, in this method, databases are partitioned in parallel to the plane of the schema diagram, which can be considered as a horizontal plane. One important feature of this approach is that each partition preserves a complete schema structure [9]. In addition, since the horizontal partition groups occurrences connected through the access path, the related occurrences will be retrieved and cached together in the buffer. Therefore, the navigation among related occurrences will be efficient.

There are a few different partition methods in object-oriented databases [7]. One of the methods is to group objects that are connected (referenced) by OID's in the aggregation hierarchy. This method can be valid because the aggregation hierarchy is equivalent to the relationship through which tables are joined in relational databases. Another method is to group all objects in the class inheritance hierarchy. In object-oriented databases, the schema graph is used to describe the data structure. The query graph for a user query is a subset of a schema graph. The user query selects all objects that satisfy the Boolean predicates on a query graph. The selection criteria contains aggregation hierarchies. The objects in an aggregation hierarchy are accessed together. The inheritance hierarchy is not the relationship in which data are accessed together but the conceptual relationship between a superclass and a subclass. Therefore, the inheritance hierarchy has less effect upon the partition than the aggregation hierarchy.

In this paper, we present a method to partition a set of objects into subsets that have objects connected by the object identifier through an aggregation hierarchy, when all the objects in a class inheritance hierarchy are initially stored in one partition block. This type of partition is similar to the horizontal partition of network or hierarchical databases. The proposed partition algorithm offers an automatic partition for object-oriented databases that results in an effective clustering of related objects.

We define a condensed schema graph as the one in which all the classes on a class inheritance hierarchy are condensed into one node, as illustrated in Figure 1.

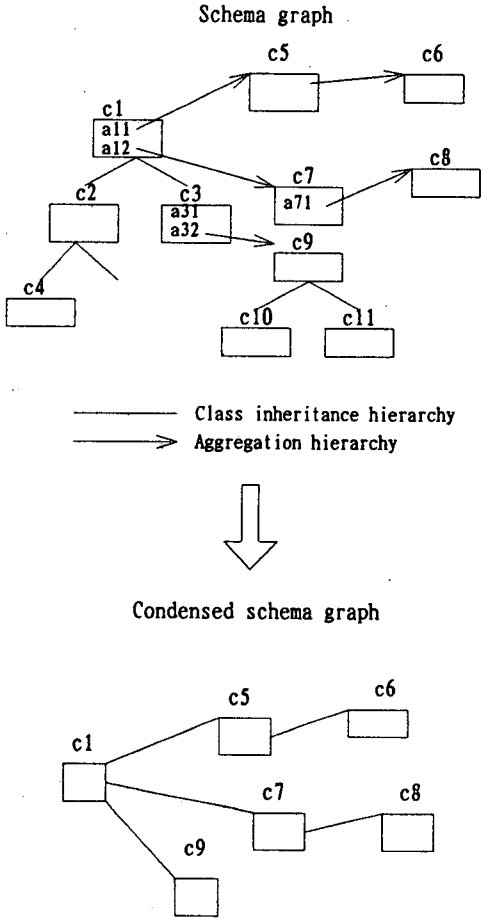


Fig. 1. Condensed schema graph.

Therefore, a condensed schema graph does not have inheritance hierarchies. The graph has aggregation hierarchies only. The objects in a schema graph are connected by OID's through aggregation hierarchies. The OID pointing direction is not important for the partition purpose because the backward reference can always be implemented. Consequently, a condensed schema graph is an undirected graph.

In section II, the set function model is explained. In section III, the algorithm that partitions object-oriented databases is described.

II. Set Function Model

1. Set function

In this section, we define a set function, an accessibility

function and an accessibility relation. We have chosen a set function for the following reasons: (1) An inverse is always defined for the set function. (2) An accessibility relation is usually required between sets of objects. (3) A set function can easily be implemented in terms of data manipulation operations. We introduce the concept of the set function and then develop its properties necessary here.

A set function is induced from a function. Let f: $X \mapsto Y$, then f induces a set function $F:2^X \mapsto 2^Y$ and its inverse $F^{-1}:2^Y \mapsto 2^X$ such that $F(D)=\{y \in Y \mid x \in D \wedge y = f(x)\}$, $F^{-1}(R) = \{x \in X \mid f(x) \in R\}$, $\forall D \in 2^X$, $\forall R \in 2^Y$, where 2^X denotes the power set of X.

In network databases and hierarchical databases, we can define a function from a set of member records to a set of owner records because there is a one-to-many relationship between owner records and member records. However, in object-oriented databases, we cannot define a function between two classes in an aggregation hierarchy because their relationship is generally many-to-many. We redefine a set function more generally.

Definition 1: Let d,r be class nodes in a condensed schema graph such that r is a domain class node of an attribute of a node d in an aggregation hierarchy. Then we denote $d \Rightarrow r$. Let s,t be class nodes in a condensed schema graph. If either $t \Rightarrow s$ or , $s \Rightarrow t$ then we denote $s \sim t$. Let x,y be objects in class nodes d,r respectively. If an attribute of x has the object identifier of an object y or a set of object identifiers one of which is that of an object y, then we write $x \Rightarrow y$. Let a,b be objects. If either $a \Rightarrow b$ or $b \Rightarrow a$, then we denote $a \sim b$.

Definition 2: Let d,r be classes such that $d \sim r$. Let D,R be sets of objects in d,r respectively. A set function $F:2^D \mapsto 2^R$ is defined such that $F(A)=\{y \in R \mid x \in A, x \sim y\}$, for any $A \subseteq D$. And a set function $F^{-1}:2 \mapsto 2^D$ is defined such that $F^{-1}(B)=\{x \in D \mid y \in B, x \sim y\}$, for any $B \subseteq R$. We denote F^{-1} an inverse set function of F.

Lemma 1: Let F be a set function such that $F:2^D \mapsto 2^R$. For any D_1, $D_2 \subseteq D$ and R_1, $R_2 \subseteq R$,

(i) $F(D_1 \cup D_2) = F(D_1) \cup F(D_2)$
(ii) $F(D_1 \cap D_2) = F(D_1) \cap F(D_2)$
(iii) $F^{-1}(R_1 \cup R_2) = F^{-1}(R_1) \cup F^{-1}(R_2)$
(iv) $F^{-1}(R_1 \cap R_2) = F^{-1}(R_1) \cap F\text{-}1(R_2)$

Proof: Clear from Definition 2.

When X and Y are finite, Lemma 1 may be extended to an arbitrary number of unions and intersections of subsets of X and Y.

Definition 3: Let $F:2^Y \mapsto 2^Z$ be a set function and $G:2^X \mapsto 2^Y$ be the same. Then the composition of set functions $FG(D)=F[G(D)]$ for all $D \subseteq X$.

The composition of any number of set functions and inverse set functions is also defined from definition 3. Using definition 3, the property of set function shown in Figure 2 is $F^{-1}F(A) \supseteq A$ and $FF^{-1}(B) \supseteq B$.

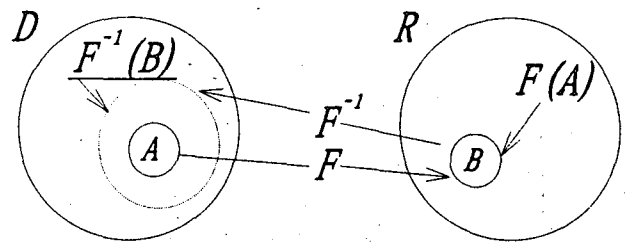


Fig. 2. Set function and inverse set function.

For notational compactness, the composition $F_{i_1}F_{i_2}\ldots F_{i_n}F_{i_1}$ is defined to be CF_{i_1,i_n} where $\{i_j,\ 1 \le j \le n\}$ is a monotonous increasing or decreasing sequence of integers with $i_{j+1} = i_j+1$ or $i_{j-1} = i_j - 1$ respectively.

Lemma 2: If F_i is a set function such that $F_i:2^X \mapsto 2^{X_{i+1}}$ for i=1,2,...,n, then for $CF_{n,1}$ and $D_1,D_2 \subseteq X_1$

Proof:*(i)* The mathematical induction is used. For n=1, from the lemma 1*(i)*. and *(ii)*, $F_1(D_1 \cup D_2)=F_1(D_1) \cup F_1(D_2)$. Suppose it is true for $n=k\text{-}1$, then $CF_{k-1,1}(D_1 \cup D_2)=CF_{k-1,1}(D_1) \cup F_{k-1,1}(D_2)$. For $n=k$,

$$\begin{aligned} CF_{k,1}(D_1 \cup D_2) &= F_k CF_{k-1,1}(D_1 \cup D_2) \\ &= F_k [\, CF_{k-1,1}(D_1 \cup D_2) \,] \\ &= F_k [\, CF_{k-1,1}(D_1) \cup CF_{k-1,1}(D_2) \,] \\ &= F_k [\, CF_{k-1,1}(D_1) \,] \cup F_k [\, CF_{k-1,1}(D_2) \,] \\ &= F_k CF_{k-1,1}(D_1) \cup FCF_{k-1,1}(D_2) \\ &= CF_{k-1,1}(D_1) CF_{k,1}(D_2). \end{aligned}$$

(ii) The same procedure as *(i)* is used. Q.E.D.

From Lemma 2, the following general formula is obtained.

Corollary 1: Let F_i be defined as in Lemma 2. For $D_j \subseteq X_1$ for j=1,...,m

(i) $CF_{n,1}(\cup_{j=1}^{m} D_j) = \cup_{j=1}^{m} CFn,_1(D_j)$
(ii) $CF_{n,1}(\cap_{j=1}^{m} D_j) = \cap_{j=1}^{m} CFn,_1(D_j)$

We define an accessibility relation with a set function as follows.

Definition 4: Let D be a set of the objects in a node of a condensed schema graph. $\rightarrow \subseteq D \times D$ is the relation such that $d_i \rightarrow d_j$ if *(i)*$i=j$ or *(ii)* there exist $CF_{l,m}$ with $d_j \in CF_{l,m}(\{d_i\})$ for $d_i,d_j \in D$. $\rightarrow$ is said to be an accessibility relation. If $d_i \rightarrow d_j$, d_j is said to be accessible from d_i.

Theorem 1: $\rightarrow$ is an equivalence relation.

Proof: *(i)* Symmetry : Suppose $x_i \rightarrow x_j$. Let $x_i \rightarrow x_{n+1}$. Then there exists a sequence $\{F_k\}$ such that $F_k: 2^{X_k} \mapsto 2^{X_{k+1}}$ for $1 \le k \le n$ where X_k is a set of objects in a node of a condensed schema graph with $x_1 \in X_1$, $x_{n+1} \in X_{n+1}$ and $x_{n+1} \in CF_{n,1}(\{x\})$. Let F_k^{-1} be an inverse set function of F_k, then there exists $CF_{1,n}^{-1}$ such that $x_1 \in CF_{1,n}^{-1}(\{x_{n+1}\})$. Therefore $x_{n+1} \rightarrow x_1$. This implies $x_j \rightarrow x_i$.

(ii) Transitivity : Suppose $x_i \rightarrow x_j$ and $x_j \rightarrow x_m$. There exists sequences $\{F_p \mid 1 \le p \le r\}$, $\{F_q \mid r+1 \le q \le s\}$ with

(a) $F_p : 2^{X_p} \mapsto 2^{X_{p+1}}$, $1 \le p \le r$

(b) $F_q : 2^{X_r} \mapsto 2^{X_{r+1}}$, $r+1 \leq q \leq s$

(c) $x_i \in X_l$, $x_m \in X_s$, $x_j \in X_{r+1}$

(d) $x_j \in CF_{r,l}(\{x_i\})$, $x_m, \in CF_{s,r+1}(\{x_j\})$

obviously, $x_m \in CF_{s,l}(\{x_i\})$. So, $x_i \rightarrow x_m$.

(iii) Reflexivity : By definition 4 $x_i \rightarrow x_i$. Q.E.D.

Theorem 1 says that an accessibility relation is reflexive, symmetric, and transitive . In addition, since an accessibility relation is an equivalence relation, an accessibility relation induces a partition such that any object is accessible by an object in the same partition block.

Definition 5: Let D be a set of the objects of a node in a condensed schema graph. $\leftrightarrow \subset D \times D$ is a relation such that $d_i \leftrightarrow d_j$ if $d_i \rightarrow d_j$ and for all $d_i, d_j \in D$. $\leftrightarrow$ is said to be a mutual accessibility relation.

If $d_i \leftrightarrow d_j$, d_i and d_j are said to be mutually accessible.

Corollary 2: $\rightarrow = \leftrightarrow$.

Proof: If part is trivial. For only if part: If $d_i \leftrightarrow d_j$, by theorem 1, $d_j \rightarrow d_i$, Hence $d_i \leftrightarrow d_j$

consequently, $\leftrightarrow$ is also an equivalence relation.

It is clear that the subsets of objects induced by the mutual accessibility relation $\leftrightarrow$ are the partition blocks of related objects.

2. *Accessibility Function*

When $x_i \rightarrow x_j$, where x_i, x_j, are the objects of the same node in a condensed schema graph, it is often the case that there exists a sequence $\{x_k | 1 \leq k \leq n\}$ such that $x_i \rightarrow x_l$, $x_k \rightarrow x_{k+1}$ for $1 \leq k \leq n-1$ and $x_i \rightarrow x_j$. We want to characterize the situation that $x_i \rightarrow x_j$ without such sequence $\{x_k | 1 \leq k \leq n\}$. It is denoted $x_i \xrightarrow{o} x_j$.

Definition 6: Let X_i be the set of objects of a node in a condensed schema graph for $1 \leq i \leq \underline{n}+1$ and $F_i: 2^X \rightarrow 2^X$. A composition $A=CF_{n,l}$ is said to be an accessibility function if $X_l=X_{n+1}$ and $X_i \neq X_1$ for $2 \leq i \leq n$. The set of all accessibility functions defined for a node in a condensed schema graph is denoted as $\mathcal{A}$.

Lemma 3: Let be the set of objects in a node in a condensed schema graph and $\mathcal{A}$ be the set of all the accessibility functions defined for C. For $x_i, x_j \in X$ $x_i \xrightarrow{o} x_j$ if and only if for some $x_j \in A(\{x_j\})$ for some $A \in \mathcal{A}$.

Proof: Clear from definition 6.

Previously, we have shown that the relation $\leftrightarrow$ induces a partition for a database. For the set of objects of a node, $\leftrightarrow$ induces a partition. The next theorem is the basis for a method to find a partition using a set function.

Theorem 2: Let S be the set of objects of a node C in a condensed schema graph and $\mathcal{A}=\{A_p | 1 \leq p \leq n\}$ be the set of accessibility functions defined for C. $X \subseteq S$ is a maximal set such that $x_i \rightarrow x_j$ for all $x_i, x_j \in X$ if and only if X is a minimal set such that $A_p(X) \subseteq X$ for all p.

Proof:($\Rightarrow$) Suppose $A_p(X) \not\subseteq X$ for some p. Let Ap(X)-X=X' $\neq \phi$. There must be $x_r \in X$ and $x_s \in X'$ such that $x_s \in A_p(\{x_r\})$. From Lemma 3, $x_r \rightarrow x_s$. Then from Theorem 1, $x_s \rightarrow x_r$. Hence $x_r \leftrightarrow x_s$. This is a contradiction to the maximality of X . So $A_p(X) \subseteq X$ for all p. Now suppose there exists $X'' \subset X$ such that $A_p(X'') \subseteq X''$ for all p. Let $X'''=X-X''$. Since $x_i \rightarrow x_j$ for all $x_i, x_j \in X$ there exist $x_{k_1} \in X'', x_{k_2} \in X'''$ such that $x_{k_1} \rightarrow x_{k_2}$. There is a sequence $x_{k_2}, \cdots, x_k, x_{k+1}, \cdots, x_{k_1}$, where $x_k \in X''$ for $2 \leq l \leq i$ and $x_k \in X'''$ for $i+1 \leq n-1$ such that $x_{k_1} \xrightarrow{o} x_{k_2} \xrightarrow{o} \cdots \xrightarrow{o} x_{k_1} \xrightarrow{o} \cdots \xrightarrow{o} x_{k_2}$. In case there is no such sequence between x_{k_1} and x_{k_2}, that is $x_{k_1} \xrightarrow{o} x_{k_2}$, we can consider $x_{k_1}=x_{k_1}$ and $x_{k_2}=x_{k_2}$. From Lemma 3, $x_{k_2} \in A_q(\{x_{k_1}\})$ for some q. So, $A_q(X'') \not\subseteq X''$, which is a contradiction. This proves the minimality.

($\Leftarrow$) Let $[x]$ be a block containing x in the partition induced by the equivalence relation $\leftrightarrow$. Suppose $x_k \leftrightarrow x_l$ for some $x_k \in X$ and $x_l \notin X$. As shown previously, $A_q(X) \not\subseteq X$ for some q. Therefore, for any $x_k \in X$ and $x_l \notin X$, $x_k \not\rightarrow x_l$. Suppose $x_i \not\rightarrow x_j$ for some $x_i, x_j \in X$. Since $\leftrightarrow$ is an equivalence relation, $[x_i] \cap [x_j] = \phi$. $[x_i] \subset X$, since $x_i \in X$ and $x_i \not\rightarrow x_j$ for any $x_l \notin X$. Let Y be a maximal set, where $[x_i] \subseteq Y \subset X$, such that $x_i \leftrightarrow x_j$ for all $x_i, x_j \in Y$. From only if part of the theorem, $A_p(Y) \subseteq Y$ for all p. This contradicts the minimality of X. Therefore, $x_i \leftrightarrow x_j$ for all $x_i, x_j \in X$. Since $x_k \not\rightarrow x_l$ for any $x_k \in X$ and $x_l \notin X$, X is a maximal set. Q.E.D.

From Theorem 2, a maximal set in which any two objects are mutually accessible can be obtained by identifying and using applicable accessibility functions.

3. *An Example for the Usage of Set Functions and Accessibility Functions*

A partition method by the set function and the accessibility function is shown. The set function is the operation primitive for the static partition algorithm. We show the set function algorithm as follows:

procedure *F(a)* : Determine the set of objects, $\beta \in 2^Y$, which are connected to one of the objects in $a \in 2^Y$, $F: 2^X \mapsto 2^Y$

```
begin
1) β ← { }
2) for all a∈a
3)       for all b ∈ Y such that a~b
4)             β ←union (β,{b})
5) return(β)
end
```

procedure $F^{-1}(\beta)$ can be defined similarly as *F(a)*. In

Figure 3, if $a=\{a,b,c\}$, then $\beta=F(a)=\{g,h,i,j\}$. If $\beta=\{j,k,l\}$, then $a=F^{-1}(\beta)=\{c,d,e,f\}$.

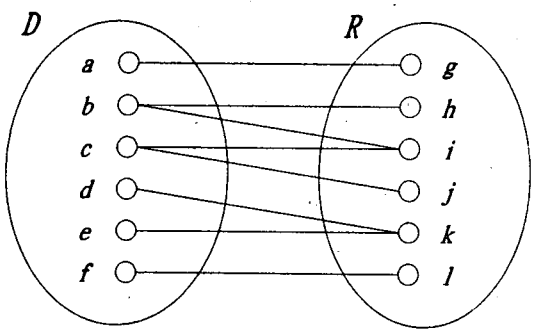


Fig. 3. $F:2^D \mapsto 2^R$

Theorem 2 suggests the static partition algorithm. It says that the minimal subset X such that $A_p(X)\subseteq X$ for all accessibility function A^p is the maximal subset for a partition.

We show an example to find maximal subsets for the partition of a graph that has only 2 nodes. In Figure 3. The accessibility function of D is $F^{-1}F$. Let $R_F=F^{-1}F$. $\{a\}$ is the maximal subset for the partition of D that satisfies $A_p(X)\subseteq X$ because $R_F(\{a\})=\{a\}$. However, $\{b\}$ is not the maximal subset for the partition of D that satisfies $A_p(X)\subseteq X$ because $R_F(\{b\})=\{b,c\}$ and $R_F(\{b\})\not\subseteq\{b\}$.

{b,c} is maximal subset for partition of D because $R_F(\{b\})=\{b,c\}$. Likewise, $\{d,e\}$ and $\{f\}$ are the maximal subsets for partition of D because $R_F(\{d,e\})=\{d,e\}$ and $R_F(\{f\})=\{f\}$. Therefore, D is partitioned into $\{\{a\},\{b,c\},\{d,e\}\{f\}$. In this example, we see the important characteristics of the accessibility function. That is, $R_F(X)=R_F^m(X)$ and $R_F(X)\subseteq X$ where. $R_F^m=R_FR_F....R_FR_F$. Using these characteristics, we can get the maximal subset for the partition of a node by applying R_F repeatedly to the subset that has an element until $R_F^m(X)=R_F^{m+1}(X)$.

To improve the efficiency, we can partition R as well as D simultaneously. If $F(A)=B,F^{-1}(B)=A$, then $F(A)=FF^{-1}(B)=B$ and $F^{-1}(B)=F^{-1}F(A)=A$. That is, A and B are the maximal subsets for the partition of D,R respectively. For example, $F(\{a\})=\{g\}$ and $F^{-1}(\{g\})=\{a\}$. This implies that $F^{-1}(\{g\})=F^{-1}F(\{a\})=\{a\}$ and $F(\{a\})=FF^{-1}(\{g\})=\{g\}$. Therefore $\{a\}$ and $\{b\}$ are the maximal subsets for a partition. Also $F(\{a\})=\{h,i\}$ and $F^{-1}(\{h,i\})=\{b,c\}$. We can apply a set function again. That is, $F(\{b,c\})=\{h,i,j\}$ and $F^{-1}(\{h,i,j\})=\{b,c\}$ and $F(\{b,c\})=FF^{-1}(\{h,i,j\})=\{h,i,j\}$ and $F^{-1}(\{h,i,j\})=F^{-1}F(\{b,c\})=\{b,c\}$. Therefore $\{b,c\}$ and $\{h,i,j\}$ are the maximal subsets for partition. Finally the partitions of D,R are $\{a,g\},\{b,c,h,i,j\},\{d,d,k\},\{f,l\}$.

III. Static Partition Algorithm

In this section , we present a static partition algorithm that partitions an object-oriented database by using accessibility function.

Lemma 4: Let a,b,c be class nodes in a condensed schema graph. Let A,B,C be the object sets of a,b,c respectively. If $a\sim b$ and $b\sim c$ and $F:2^A\mapsto 2^B$, $G:2^B\mapsto 2^C$, then there exist the natural numbers n,m such that $F^{-1}R_G^nFR_F^m(\{a\})=X$, for all $a\in X$ and $R_F^m(\{s\})=R_F^{m+1}(\{s\})$ and $R_G^n(T)=R_G^{n+1}(T)$ and $T=FR_F^m(\{a\})$ for a minimal subset such that $F^{-1}G^{-1}GF(X)\subseteq X$ and $X\subseteq A$

Proof: Let m be the smallest natural number such that $R_F^m(\{a\})=R_F^{m+1}(\{a\})$. Let $FR_F^m(\{a\})=Y$, $Y\subseteq B$. Then $F^{-1}(Y)=F^{-1}FR_F^m(\{a\})=R_F^m(\{a\})$. Let n be the smallest natural number such that $R_F^n(Y)=R_F^{n+1}(Y)$. Let $R_F^n(Y)=Y'$ and $G(Y')$ and $Z\subseteq C$. then $G^{-1}(Z)=G^{-1}GR_G^{n+1}(Y)=R_G^{n+1}(Y)=R_G^n(Y)=Y'$. Let $X=F^{-1}R_G^nFR_F^m(\{a\})$. Clearly $a\in X$ and $X=F^{-1}R_G^nFR_F^m(\{x\})$ for all $x\in X$. By Corollary 1, $F^{-1}R_G^nFR_F^m(\{X\})=X\cup X\cup...\cup X=X$.

And $X=F^{-1}R_G^nFR_F^m(X)=F^{-1}\underbrace{(G^{-1}G)...(G^{-1}G)}_{n}\underbrace{F(F^{-1}F)...(F^{-1}F)}_{m}(X)$

$=F^{-1}G^{-1}G\underbrace{(G^{-1}G)...(G^{-1}G)}_{n-1}\underbrace{(FF^{-1})...(FF^{-1})}_{m}F(X)\supseteq F^{-1}G^{-1}GF(X).$

So, $X\supseteq F^{-1}G^{-1}GF(X)$. For $X'\subset X$, $F^{-1}R_G^nFR_F^m(X')=X\cup X\cup...\cup X=X$. If p and q are the smallest natural numbers such that $F^{-1}R_G^pFR_F^q(X')$, then $p<n$ or $q<m$ by the property of a set function like $FF^{-1}(A)\supseteq A$. If $q<m$, that is, $m>1$, then $R_F(\{b\})=F^{-1}F(\{b\})\supset\{b\},b\supset X'$. Therefore, $F^{-1}G^{-1}GF(X')\supseteq F^{-1}F(X')\supset X'$. If $p<n$, that is, $n>1$, then $G^{-1}G(K)\supset K,K\subseteq B$, so, $F^{-1}G^{-1}GF(X')\supseteq F^{-1}F(X')\supset X'$. Therefore, X is a minimal subset such that $F^{-1}FGF(X)\subset X$. Q.E.D.

Lemma 4 shows a partition method for the objects of three nodes connected in serial. If $F^{-1}FR_F^m(a)=R_G^{m+1}(a)=R_F^m(a)$ and $P=R_F^m(a)P\subseteq A$, then $F^{-1}F(P)=P$. And if $FR_F^m(a)=T,T\subseteq B$,then $FF^{-1}FR_F^m(\{a\})=FR_F^{m+1}(a)=FR_F^m(a)$. So $FF^{-1}(T)=T$. That is, P,Q are the maximal subsets for the partition of A and B. Again if $G^{-1}GR_G^n(T)=R_G^{n+1}(T)=R_G^n(T)$ and $R_G^n(T)=V,V\subseteq B$, then $G^{-1}G(V)=V$. Also if $G(V)=W,W\subseteq C$, then $GG^{-1}(W)=W$ because $GG^{-1}G(V)=G(V)$. That is V and W are the maximal subsets for the partition of B and C. B and C are partitioned completely but A is not partitioned completely. A is partitioned by $F^{-1}R_G^n(T)=F^{-1}R_G^nFR_F^m(a)$.

Now we present a static partition algorithm that partitions an object-oriented database by searching a condensed schema graph using depth first search. The input of this algorithm is a condensed schema graph and its object graph. The output is the partition blocks that have all objects connected by object identifiers.

This algorithm starts from an arbitrary node in an undirected condensed schema graph and searches the graph using depth first search and finds a maximal subset for the partition of each node. Procedure Partition is the basic algorithm that partitions 2 nodes v,w in edge (v,w). Searching the graph using depth first search generates a spanning tree without cycle edges. We can apply the method shown in Lemma 4 repeatedly to all tree edges. The algorithm continues to partition nodes in the tree edges until it meets a terminal node of the spanning tree. When the algorithm meets the terminal node, it backtracks to all visited nodes and partitions again the two nodes adjacent to an edge. As a result, several subsets are generated in each visited node of the graph. We call them N-partition in this algorithm. The collection of the N-partitions that are connected by object identifiers but not in the same node is the total partition we want to find. We call this T-partition. After this algorithm partitions all the tree edges, it is applied to the remaining cycle edges (i.e. back edges) in the graph. Each node in the cycle edge is already partitioned before partitioning the nodes in cycles edges. If some N-partitions are merged during the partitioning of the nodes of a cycle edge, the corresponding T-partitions are also merged for the merged N-partitions. Algorithm Static_Partition and Procedure Partition are described below.

Algorithm Static_Partition (G: condensed schema graph)

1. Generate a spanning tree from the condensed schema graph using depth first search.
2. Search the spanning tree using depth first search until a terminal node is met and apply Procedure Partition to the visited nodes adjacent to an edge.
3. When a terminal node is met, backtrack to all visited nodes and partition again the nodes adjacent to an edge.
4. Merge the connected N-partitions into the T-partition.
5. Repeat 2, 3 and 4 until all nodes of the tree edges are partitioned
6. For each cycle edge, apply the Procedure Partition to the nodes in the cycle edge.
   For merged N-partitions in a node, merge the T-partitions that have the merged N-partitions.
   For connected N-partitions between two nodes, merge the T-partitions that have the merged N-partitions.

Procedure Partition (E: edge)

/* E=(v,w),V,W are the set of objects in nodes v,w respectively. */
/* $F:2^V \mapsto 2^W$: a set function between the nodes in the edge E=$(v,w,)$ */
/* All subsets $A \subseteq V$, $B \subseteq W$ that satisfy $F(A)=B, F^{-1}(B)=A$ are called N-partition. */
1. Find all pairs (A,B) such that $F(A)=B, F^{-1}(V)=A, A \subseteq V, B \subseteq W$ by applying set functions F and F^{-1} repeatedly.

We show an example for the algorithm. We get a spanning tree from a condensed schema graph as shown in Figure 4.

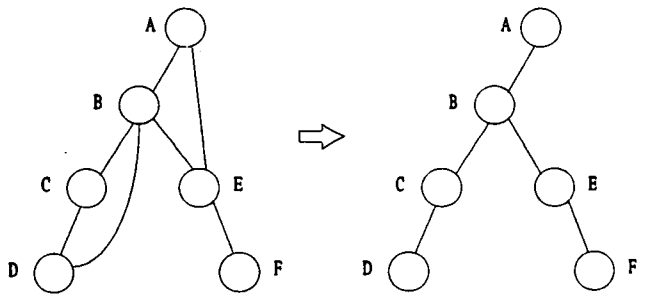


Fig. 4. Condensed schema graph and its spanning tree.

Figure 5 shows an example of the object graph of the condensed schema graph. The algorithm starts from A node. It partitions edges (A,B), (B,C) and (C,D) sequentially. D is a terminal node (Figure 6 (a)). It backtracks and partitions again (C,B) and (B,A) (Figure 6 (b)). It searches the remaining tree edges and partitions (B,E) and (E,F). F is a terminal node (Figure 6 (c)). It backtracks and partitions again (E.B), (B,C), (C,D), (B,A) (Figure 6 (d)). All nodes in the tree edges are partitioned. The nodes of the cycle edge, (B,D), are partitioned. Two T-partitions are merged (Figure 6 (e), Figure 6 (f)). The nodes of the other cycle edge (A,E) are partitioned. Two T-partitions are merged (Figure 6 (g), Figure 6 (h)).

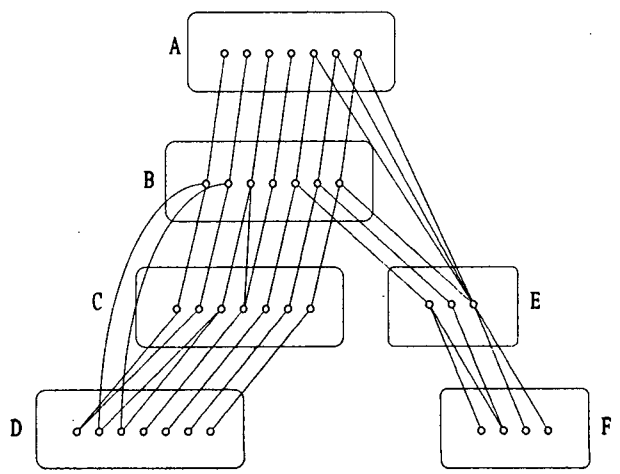


Fig. 5. Object graph of the condensed schema graph

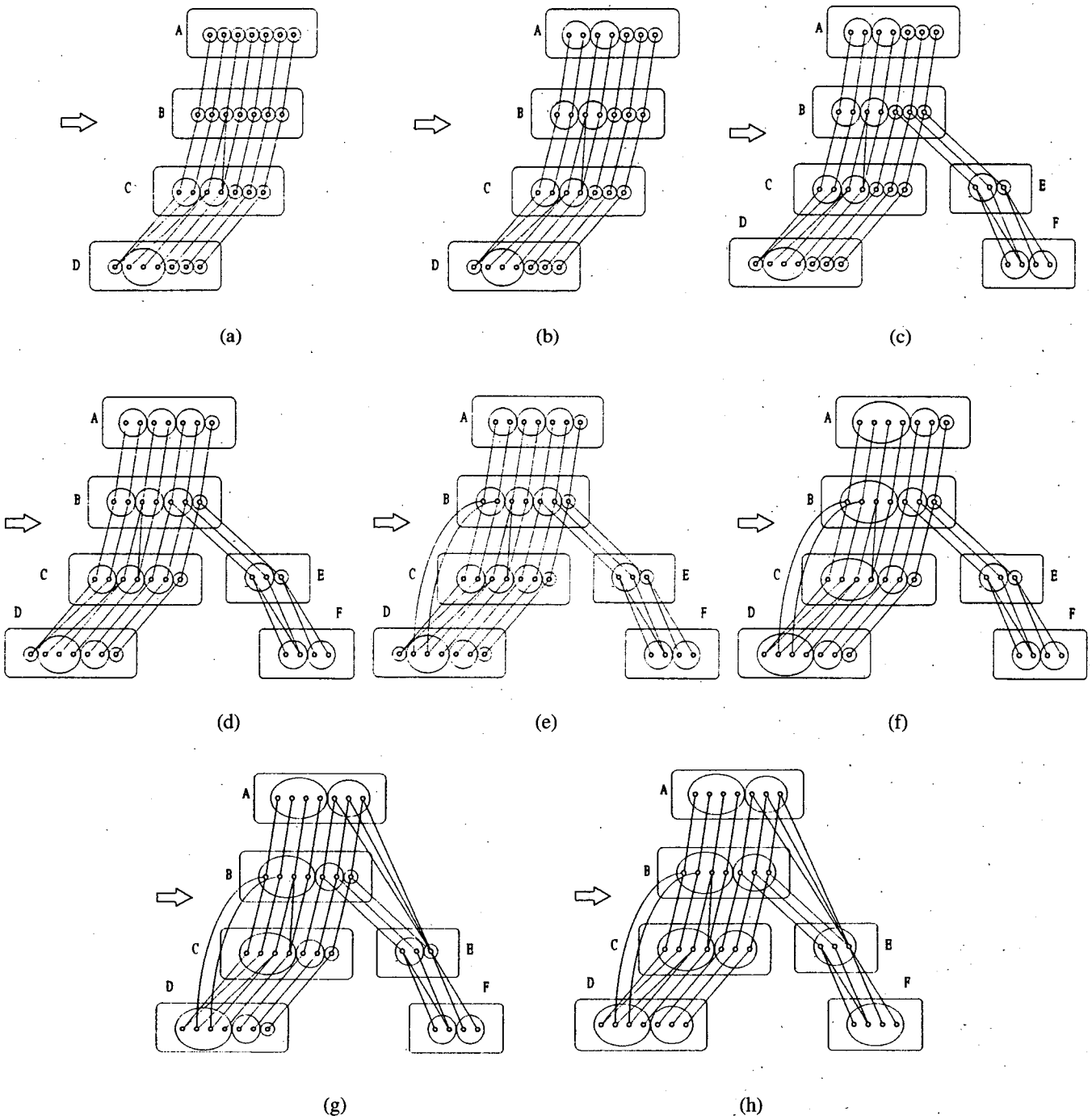


Fig. 6. Partition steps using algorithm Static_Partition.

The partition method presented in this paper can be applied to databases that have access paths between data, such as network databases and hierarchical databases as well as object-oriented databases. In fact, the hierarchical DBMS[5] such as IMS uses a similar method which preserves the schema structure in each partition block.

IV. Conclusion

This paper presented a formal method to find a parti-tion that are induced by the aggregation hierarchy in object-oriented databases. However, this method can be applied to other types of databases that have access paths

between data. The formal method includes the set function modeling of databases that have explicit access paths such as network databases, hierarchical databases and object-oriented dababases. In addition, the method provide an accessibility function and an accessibility relation to find a maximal subset for partitioning databases. This method is a static method that derives an initial partition of a database. The dynamic repartition is needed as the database is changed.

V. Acknowledgment

This research was supported in part by the grant from '93 Multimedia Information Systems Platform Development Project of the Center for Artificial Intelligence Research.

References

[1] J. Banerjee, W. Kim, S.J. Kim, and J.F. Garza, "Clustering a DAG Databases," *IEEE Transactions on Software Engineering* , Vol. 14, No. 11, Nov. 1988.

[2] E. Bertino and L. Martino, "Object-Oriented Database Management Systems : Concepts and Issues," *Computer*, Vol. 24, No. 4, pp. 33-47, April 1991.

[3] E.E. Chang, and R.H. Katz, "Exploiting Inheritance and Structure Semantics for Effective Clustering and Buffering in an Object-oriented DBMS," *Proc. of ACM SIGMOD Conference*, pp. 348-357, 1989.

[4] J.R. Cheng,. and A.R. Hurson, "Effective clustering of complex objects in object-oriented databases," *Proc. of ACM SIGMOD Conference*, pp. 22-31, 1991.

[5] C.W. Chung and K.E. McCloskey, "Access to Indexed Hierarchical Databases Using a Relational Query Language," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No.1, pp. 155-161, February 1993.

[6] K. Hua and C. Tripathy, "Object Skeletons:An Efficient Navigation Structure for Object-Oriented Database Systems," *Proc. of the International Conference on Data Engineering*, pp. 508-517, February. 1994.

[7] W. Kim, *Introduction to Object-Oriented Databases*, p. 113, The MIT Press, 1991.

[8] W. Kim, J.F. Garza, N. Ballou, D. Woelk, "Architecture of the ORION Next-Generation Database System," *IEEE Transactions on Knowledge and Data Engineering*, Vol.2, No.1, pp. 109-124, March 1990.

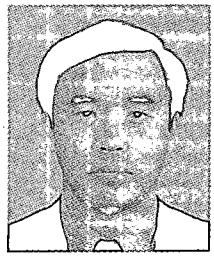
[9] Y.E. Lien and J.H. Ying, "Design of a Distributed Entry-Relationship Database System," *Proceedings of COMPSAC*, 1978.

[10] E. Omiecinski, and P. Scheuermann, "A Parallel Algorithm for Record Clustering," *ACM TODS*, Vol.15, No.4, pp. 599-624, December 1990.

[11] M.T. Ozsu, *Principles of Distributed Database System*, pp. 562, Prentice Hall, 1991.

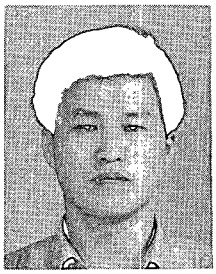
[12] M.M. Tsangaris, and J.F. Naughton, "A Stochastic Approach for Clustering in Object Bases," *Proc. of ACM SIGMOD Conference*, pp. 12-21, 1991.

[13] C.T. Yu, C.M. Suen, K. Lam, and M.K. Siu, "Adaptive Record Clustering," *ACM TODS*. Vol.10, No.2, pp. 180-204, June 1985.



Chin-Wan Chung received the B.S. degree in electrical engineering from Seoul National University in 1973 and the Ph.D. degree in computer engineering from the University of Michigan in 1983. From 1983 to 1993, he was a senior research scientist and a staff research scientist in the Computer Science Department at General Motors Research Laboratories. Since 1993, he has been an associate professor in the Department of Information and Communication Engineering at Korea Advanced Institute of Science and Technology, Seoul Campus. He is a member of the Korea Information Science Society, IEEE Computer Society and ACM. His research interests include object-oriented databases, geographic information systems, multimedia databases, distributed databases, and CIM.

Ju-Hong Lee received the B.S. degree MS degree in computer engineering from Seoul National University, Seoul, Korea, in 1983 and 1985, respectively. He was a senior software engineering at IBM. He is currently a Ph.D. candidate in the Department of Information and Communication Engineering at Korea Advanced Institute of Science and Technology, Seoul, Korea. His research interests include clustering and transaction management in distributed databases, and object-oriented databases.



Chang-Ryong Kim received the B.S. degree and M.S. degree in computer science & statistics from Seoul National University, Seoul, Korea, in 1990 and 1992, respectively. During 1992-1993, he was a research engineer at SamSung Advanced Institute of Technology. He is currently a Ph.D. candidate in the Department of Information and Communication Engineering at Korea Advanced Institute of Science and Technology, Seoul, Korea. His research interests include distributed databases, object-oriented databases, geographic information system software. He is a member of the Korea Information Science Society, IEEE Computer Society and ACM.