MAIN SCHEMA-EXTERNAL SCHEMA INTERACTION IN HIERARCHICALLY ORGANIZED DATA BASES

by

A. G. Dale
N. B. Dale

February 1977 TR-66

DEPARTMENT OF COMPUTER SCIENCES
THE UNIVERSITY OF TEXAS AT AUSTIN

MAIN SCHEMA-EXTERNAL SCHEMA INTERACTION IN HIERARCHICALLY ORGANIZED DATA BASES

A. G. Dale

N. B. Dale

Department of Computer Sciences

The University of Texas at Austin

A class of external schemas derivable from a tree structured main schema is identified. It is shown that the properties of this class of schemas permit the construction of a processing interface such that predicates defined on an external schema can be evaluated in an occurrence structure disciplined by the main schema.

Introduction

As a data base increases in size and complexity and as the number and variety of users increases, it becomes important to support a variety of user views of the logical organization of the data. This requirement is partly an intrinsic consequence of an increase in user population and in the range of information processing applications supported by the data base. It is also related to questions of security, in the sense that parts of the data base may have to be masked from particular user or application views.

The mechanisms for supporting this requirement can be constructed in the context of a data base management system with a two-schema architecture. In such systems there will exist a main schema for the global data base and a set of external schemas relevant to particular user views of its logical structure.

In the present paper we consider the particular case of a system where both the main schema and the external schemas are trees of record types - i.e.

hierarchically organized systems. Associated with the main schema is an occurrence structure of record instances, logically partitioned by level and by record type, consistent with the partitioning defined in the main schema. We assume that an external schema is a transformation of the main schema involving some hierarchical rearrangement of record types. We do not consider other possible transformations (i.e. data item renaming, data type transformations, intra-record transformations, or the introduction of new record types into the external schema). We address the following problem:

Given a main schema S and an associated occurrence structure 0 what configurations of an external schema S_E are permissable, such that processes defined with respect to S_E will successfully execute on 0.

It is evident that the problem must be considered in a specific system context. In particular we assume:

1. The occurrence structure 0 is a transitive hierarchical acyclic network with (possibly) data equivalent nodes. This data model has been described fully in [1]. In the limiting case 0 will be a tree of record occurrences.
2. Data base processes are invoked in a non-procedural predicate language in statements of the general form ACTION X WHERE P. In this frame, X is a subset of record types contained in the external schema, and P is a hierarchical predicate. A hierarchical predicate is a Boolean expression defined on data element names and values, and thus also contains (at least implicitly) references to record types in the external schema, i.e. record types containing the data element names mentioned in P.
3. A condition for evaluating a predicate requires that all references to record types in P be to record types contained in a single

branch set of the external schema.[1] A branch set will be defined in the next section.

The semantics of the operations that can be invoked in a predicate are as follows:

Complementation is interpreted with respect to a universe consisting of nodes of a given type R. If T is the set of nodes of record (group) type R, and N is a set of nodes, $N \varepsilon T$, selected by a term of predicate then $\sim N = T - N$.

Intersection and union operations are realized by defining the operation on normalized sets of nodes as follows: if term 1 of P selects a set T_1 of nodes of record (group) type R_1 and term 2 of P selects a set T_2 of nodes of record (group) type R_2 and Level (R_1) < Level (R_2), then the operation is defined either with respect to T_1 and a set of nodes, T_3, such that $T_3 \varepsilon$ ANC(T_2) and T_3 contains nodes only of type R_1 or with respect to T_2 and a set of nodes T_4, such that $T_4 \varepsilon$ DESC(T_1) and T_4 contains nodes only of type R_2. See the next section for definitions of ANC and DESC.

Basic Definitions

Level: Given a tree T, if t_o is the root of T, Level $(t_o) = 0$. If t_1, $t_2 \varepsilon$ T, Level $(t_1) = k$, and t_2 is a child of t_1, then Level $(t_2) = k+1$. The level of a record (group) occurrence is the level of the record (group) in the schema tree.

Descendant set: Given a tree, T, and a node $t_i \varepsilon$ T, DESC$(t_i) = \{t \varepsilon T | t$ is a node in a tree with root $t_i\}$.

---

1. This is, for example, the condition for processing of basic Boolean predicates in SYSTEM 2000, although the predicate language of that system also incorporates features that permit inter-branch set operations.

Ancestor set: Given a tree, T, and a node $t_i \in T$, $ANC(t_i) = \{t \in T \mid t_i \in DESC(t)\} - \{t_i\}$.

Broom set: Given a tree, T, and a node $t_i \in T$, the broom set of t_i, denoted $B(t_i) = DESC(t_i) \cup ANC(t_i)$.

Branch set: If S is a tree-structured schema, with a terminal node s_i, a branch set $BR_i \in S$, is $B(s_i)$. If there are j terminal nodes, there are j branch sets in the schema.

Schema: A schema, S, is a tree such that each node corresponds to a record type.

Data Base: A data base D is a pair (S,O) where S is a tree structured (main) schema and O is an acyclic digraph such that each node $n \in O$ represents a record (group) occurrence.

Node type: If R is a record type in a schema S, a node $r \in O$ which represents an instance (occurrence) of R is said to be a node of type R.

Hierarchical predicate: A hierarchical predicate P of D is a Boolean expression defined on data element names and values, and for the binary operators (AND,OR) is of the general form:

<data element name> <relational operator> <value> <Boolean operator> <data element name> <relational operator> <value> .

For the unary operator (NOT) the expression is of the form:

<Boolean operator> <data element name> <relational operator> <value> .

Selection set: A selection set, SS(P,O) is defined to be the set of nodes of O <u>selected</u> by P, given the semantics described in (3) above.

Consistent schemas

Given a data base D = (S,0), an external schema S_E is consistent with S if every hierarchical predicate definable on S_E is computable on 0. By this we mean that given $(S_E,0)$ an algorithm exists to compute SS(P,0) unambiguously for all predicates definable on S_E. Three classes of consistent external schemas can be identified. We term these:

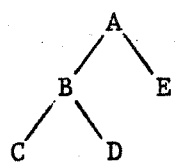
- Subschemas
- Ancestor schemas
- Descendant schemas

In each case these schemas are derivable by the application of a sequence of simple transformations on the main schema. In subsequent discussion we shall frequently refer to a schema from which other schemas are derived as the reference schema.

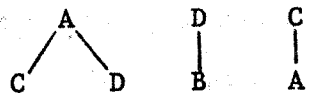
1. A subschema S_S of a reference schema S is a schema such that:
   1. The set of nodes in S_S is a proper subset of set of nodes in S.
   2. For each node s_i in S_S, the broom set of s_i in S_S is a subset of the broom set of s_i in S.

   Comment: Note that by this definition a subschema is not required to be a subgraph consisting of adjacent nodes in the reference schema. Moreover, the subschema may specify a rearrangement of nodal relationships. The following diagram illustrates the subschema construct.

Reference schema



Some subschemas

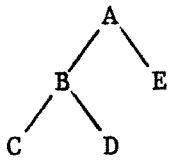


2. An <u>ancestor schema</u> S_A of a reference schema S is a schema such that:

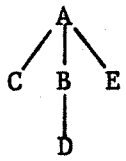
   1. The set of nodes in S_A = the set of nodes in S.
   2. For each node s_i in S_A,S, the broom set of s_i in S_A is a subset of the broom set of s_i in S.

   Comment: The reference schema S can be constructed from any of its ancestor schemas by a sequence of simple transformations, as noted later in this paper.

<u>Reference schema</u>



<u>Some ancestor schemas</u>

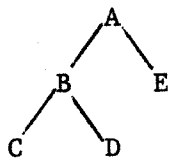


3. A <u>descendant schema</u> S_D of a reference schema S is a schema such that:

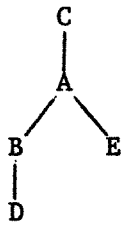
   1. The set of nodes in S_D = the set of nodes in S.
   2. For each node s_i in S,S_D, the broom set of s_i in S is a subset of the broom set of s_i in S_D.

   Comment: A descendent schema is constructed from a reference schema by a sequence of simple transformations, as shown later.

<u>Reference schema</u>



<u>Some descendant schemas</u>



It will be shown that with respect to any schema S, taken as a reference schema, that any hierarchical predicate defined on a subschema, an ancestor schema, or a descendant schema of S is computable on an occurrence structure 0, defined by S.

Predicate definition on subschemas and ancestor schemas

The definitions of a subschema and an ancestor schema preserve by construction the conditions necessary for computability on 0. In each case, broom sets in the derived schema structures are subsets of broom sets in the reference schema. Branch sets in derived schemas of these types are also subsets of branch sets in the reference schema. Consequently, the set of predicates definable on both types of derived schemas must be a subset of the set of predicates definable on the reference schema. Hence, all predicates defined on such schemas must be directly computable on 0. Clearly, the condition also holds for any subschema of an ancestor schema.

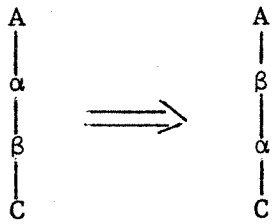
Predicates defined on descendant schemas

In a previous paper [1] we identified a primitive rearrangement transformation of tree structured schemas that preserves query homomorphism under conditions where the occurrence structure is to be mapped to a new state corresponding to a transformed schema. The constraints on the schema transformations are those specified in this paper in the definition of descendant schemas, i.e. the transformations must be such that $B(s_i)$ in $S \in B(s_i)$ in S_D for all s_i.

The primitive rearrangement operation involves the rearrangement of a hierarchically adjacent pair of nodes (α,β) such that the parent-child relationships are reversed. There are four cases to be considered. In each case the resulting schema is a descendant schema by our definition.

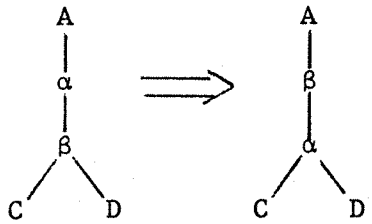
Case I

Neither node has more than one child.



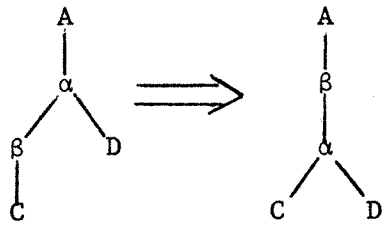
Case II

The node being moved has more than one child, but the parent node does not.



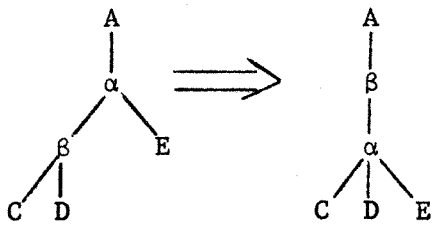
Case III

The parent of the node being moved has more than one child, but the node being moved does not.



Case IV

Both nodes have multiple children.



In the diagrams, and throughout the remainder of this paper we have adopted the convention that:

A is a record type which is the parent of record type α.
C, D, E are record types which are children α or β.
α and β are record types such that α is the parent of β.

In considering the transformations A, C, D and E may be branching, non-branching or null. α is non-branching in Case I and Case II and branching in Case III and Case IV. We term the transformation identified above the lifting transformation.

Wang [2] has introduced an additional primitive rearrangement operation which may be characterized as follows:

If there exist nodes α, β, and Ω such that:

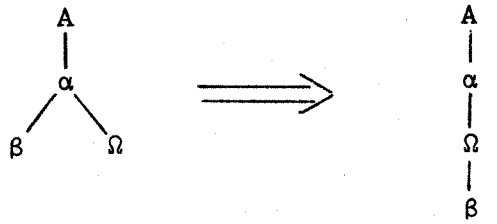
α is the parent of β.
α is in the ANC (Ω)
β is not in the ANC (Ω)

then β (or DESCβ if β is non-terminal) may be removed as a child of α and made a child of Ω. [1]

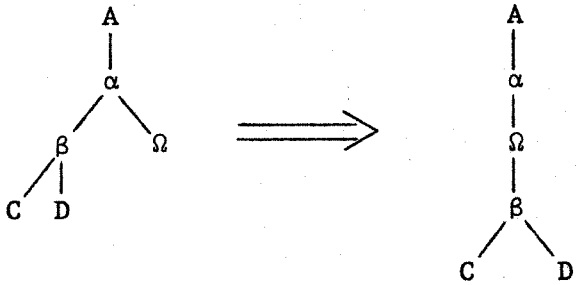
---

1. In Wang's formulation the operation is defined as a two-step transformation that inserts a node x between two nodes y and z where y is the parent of z. The first step appends x as a child of y, and the second appends z as the child of x. The two-step operation is used by Wang in an algorithm for discovering the sequence of primitive transformations necessary to generate a descendant schema from a reference schema and preserves certain properties of intermediate branch sets which facilitate the algorithm.

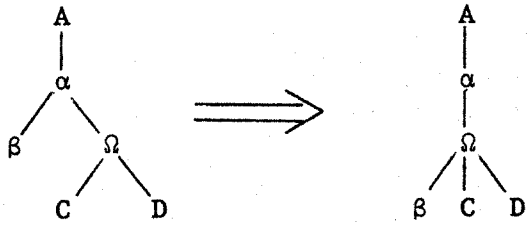
Case I. (β, and Ω are both terminal)



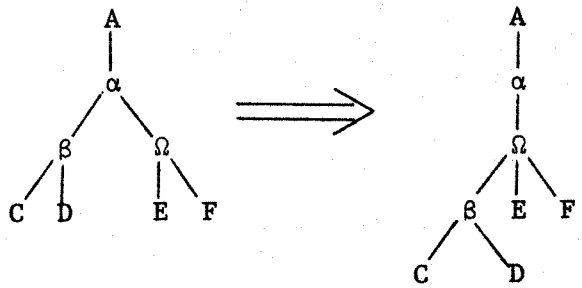
Case II. (β is branching and Ω is terminal)



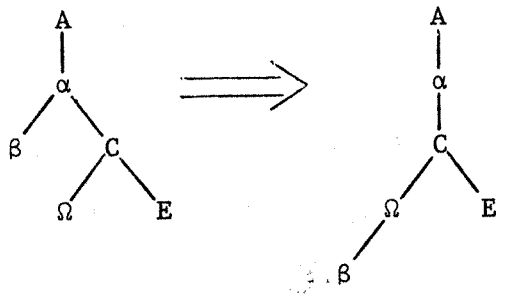
Case III. (β is terminal and Ω is branching)



Case IV. (Both β and Ω are branching)



Note that Ω need not be a child of α . The following reformulation of Case I illustrates the transformation where α is not hierarchically adjacent to Ω .



We term this transformation the shifting transformation. Wang has shown that any descendant schema can be derived by a sequence of lifting and shifting operations. In the following two tables we indicate by enumeration the results of the two transformations on the example cases outlined above for the purpose of identifying changes in broom structure that occur as a result of the transformations. As will be shown, information regarding changes in schema structure that occur as a result of applying a sequence of primitive transformations can be used by the predicate selection algorithm operating on an occurrence structure corresponding to the reference schema, when processing predicates defined against a descendant schema.

Table I

Broom structure changes under the lifting transformation

Case	Elements	Broom	
		S	S_D
I	A, α, β, C	A, α, β, C	A, β, α, C
II	A, α, β	A, α, β, C, D	A, β, α, C, D
	C	A, α, β, C	A, β, α, C
	D	A, α, β, D	A, β, α, D
III	A, α	A, α, β, C, D	A, β, α, C, D
	β	A, α, β, C	A, β, α, C, D *
	C	A, α, β, C	A, β, α, C
	D	A, α, D	A, β, α, D *
IV	A, α	A, α, β, C, D, E	A, β, α, C, D, E
	β	A, α, β, C, D	A, β, α, C, D, E *
	C	C, β, α, A	C, α, β, A
	D	D, β, α, A	D, α, β, A
	E	A, α, E	A, β, α, E *

* Broom in S not the same as broom in S_D

Table II

Broom structure changes under the shifting transformation

Case	Element(s)	Broom	
		S	S_D
I	A, α	A, α, β, Ω	A, α, β, Ω
	β	A, α, β	A, α, β, Ω *
	Ω	A, α, Ω	A, α, β, Ω *
II	A, α	A, α, β, Ω, C, D	A, α, β, Ω, C, D
	β	A, α, β, C, D	A, α, β, Ω, C, D *
	C	A, α, β, C	A, α, β, Ω, C *
	D	A, α, β, D	A, α, β, Ω, D *
	Ω	A, α, Ω	A, α, β, Ω, D, C *
III	A, α	A, α, β, Ω, C, D	A, α, β, Ω, C, D
	β	A, α, β	A, α, Ω, β *
	Ω	A, α, Ω, C, D	A, α, Ω, C, D, β *
	C	A, α, Ω, C	A, α, Ω, C
	D	A, α, Ω, D	A, α, Ω, D
IV	A, α	A, α, β, C, D, Ω, E, F	A, α, β, C, D, Ω, E, F
	β	A, α, β, C, D	A, α, Ω, β, C, D *
	C	A, α, β, C	A, α, Ω, β, C *
	D	A, α, β, D	A, α, β, Ω, D *
	E	A, α, Ω, E	A, α, Ω, E
	F	A, α, Ω, F	A, α, Ω, F
	Ω	A, α, Ω, E, F	A, α, β, C, D, Ω, E, F *

* Broom in S not the same as broom in S_D

An examination of Table I shows that the broom sets in Case I and Case II lifting moves are unchanged. Therefore predicates defined over this subclass of descendant schemas are directly computable in O. Broom and branch sets are not preserved, however, in Case III or Case IV lifting moves. In Case III record type β has been added to the branch set of D; conversely D has been added to the broom set of β. In Case IV record type β has been added to the branch set of E; and conversely E has been added to the broom set of β. In Case III and IV, the changes are thus equivalent: record type β has been added to the branch set(s) of its siblings.

An examination of Table II shows that the broom structure is changed for most nodes in every Case where the shifting transformation is applied: β and its descendants (if any) are added to $B(\Omega)$ and to the broom sets of any nodes between α and Ω. Ω is added to $B(\beta)$ and to the broom sets of the descendants of β(if any).

Consequently predicates definable on parts of a descendant schema will, in general, not be definable on S. More importantly in the present context, an algorithm designed to evaluate hierarchical predicates using the selection processes noted previously would fail to process predicates defined on those parts of a descendant schema, because the organization of the data base conforms to the structure embodied in S, that is, the main schema.

Evaluating predicates defined on external schemas

Recall that the selection process is defined on sets of nodes in the occurrence structure T_1, T_2, T_3, and T_4 where T_1 is the set selected by term 1 of a predicate and T_2 is the set selected by term 2 of the predicate. T_3 is the set of nodes of the same record type as those in T_1 (record type R_1) that are hierarchically related to the nodes in T_2. T_4 is a set of nodes of

the same record type as those in T_2 (record type R_2) that are hierarchically related to the nodes in T_1. Binary operations may be defined either on T_1 and T_3 or on T_2 and T_4. For simplicity in discussion, we will assume that the procedure is applied to T_1 and T_3. The process of selecting sets T_3 or T_4 is termed normalization.

Normalization requires that a hierarchical path exist in the occurrence structure from a node in T_2 to a node of record type R_1. This condition is guaranteed in a logical occurrence structure disciplined by its own (main) schema.

In the case of a subschema a path in the occurrence structure may be longer than the logical path in the subschema. In practice this means that the user cannot refer to record types omitted in the external view. However, hierarchical paths between occurrences of record types in the subschema exist, and direct normalization can occur.

Similarly, hierarchical paths in the occurrence structure are typically longer than the user of an ancestor schema might visualize, but direct normalization can always occur in processing predicates defined on ancestor schemas.

The difficulty with a descendant schema, as we have shown, is that the branch and broom sets in the reference schema may be expanded. In particular, application of the shifting transformation may imply a virtual hierarchical path in the occurrence structure that does not, in fact, exist. Hence, normalization cannot occur directly in 0. The paths which allow T_3 to be defined must be simulated.

Wang [2] has shown that for any reference schema S and a valid descendant schema S_D, a sequence of intermediate schemas S_1, $S_2 \ldots S_N$ exists such that S_1

is a one-step move[1] from S, and S_{i+1} is a one-step move from S_i (i=1,N and $S_D = S_{N+1}$). Therefore the transformation from S to S_D can be represented as a set of N+1 triples ($\overline{\beta}$, α, L) where :

$\overline{\beta}_i$ is a list containing β if the move is a lift and DESC (β) if the move is a shift.

α is the parent of β in the schema state preceding the move.

L_i is a list containing descendant sets in the reference schema of the siblings of β if the move is a lift, or the set of record types in S_i in the branch between α and Ω inclusive if the move is a shift.

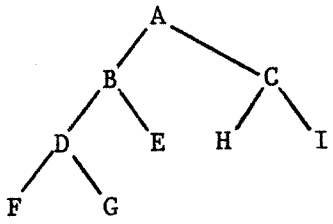
The following example shows a possible sequence of primitive transformations that would generate a target descendant schema from a main schema, and the corresponding triples describing each transformation are identified.

---

1. That is, S_1 is generated by applying either the lifting or shifting transformation to one node in S.

Example

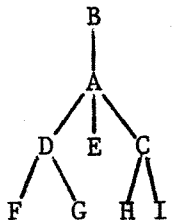
Main Schema



Lifting B

$$\begin{cases} \overline{\beta}_1 : B \\ \alpha_1 : A \\ L_1 : C, H, I \end{cases}$$

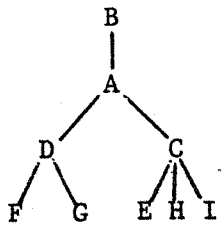
Intermediate Schema I



Shifting E

$$\begin{cases} \overline{\beta}_2 : E \\ \alpha_2 : A \\ L_2 : C, A \end{cases}$$

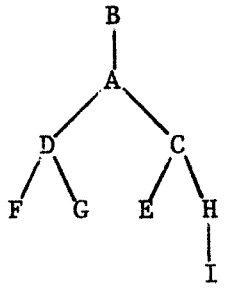
Intermediate Schema II



Shifting I

$$\begin{cases} \overline{\beta}_3 : I \\ \alpha_3 : C \\ L_3 : H, C \end{cases}$$

Target Descendant Schema



We can now describe a general selection algorithm for computing SS(P,0) for predicates defined on consistent external schemas. For the sake of simplicity we confine the discussion to binary predicates, but the procedure is generalizable to n-ary predicates. It is assumed that the processor has access to the set of triples describing the transformation sequence between reference and derived external schemas.

In [3], Parsons et. al. defined an ADJUST operation on tree structures with two input parameters S and T, where S is a set of record occurrences and T is a record type. The function returns a list of record occurrences of record type T which are related hierarchically to the input set S. The ADJUST operation realizes the normalization process previously described.

Using the ADJUST operation, intersection and union can be realized as follows in all cases. If term 1 of P selects a set T_1 of nodes of record type R_1, term 2 of P selects a set T_2 of nodes of record type R_2 then intersection and union are defined with respect to sets T_3 and T_4 where T_3 and T_4 are defined as follows:

1. If $R_1 \in \overline{\beta}_i$ and $R_2 \in L_i$ or $R_2 \in \overline{\beta}_i$ and $R_1 \in L_i$

   $T_3 = \text{ADJUST}(T_1, \alpha_i)$

   $T_4 = \text{ADJUST}(T_2, \alpha_i)$

Otherwise:

2. If the level of R_1 < the level of R_2

   $T_3 = \text{ADJUST}(T_2, R_1)$

   $T_4 = T_1$

3. If level of R_1 > the level of R_2

   $T_3 = \text{ADJUST}(T_1, R_2)$

   $T_4 = T_2$

Otherwise:

4. $T_3 = T_1$

   $T_4 = T_2$

Rule 1 above covers the case where record types referenced in a predicate are in a different hierarchical relationship in the external schema than in the main schema. In that case normalization can always be done on occurrences of the α type node associated with the simple transformation where the first hierarchical rearrangement affecting the referenced record types occurred.

Conclusion

We have identified classes of external schemas that can be derived from a rearrangement of tree-structured main schema such that hierarchical predicates definable on an external view are computable on a record occurrence structure disciplined by the main schema. We have also identified a simple procedure for evaluating predicates defined on such external views.

Topics covered in this paper provide the framework for the development of a processing interface for a class of hierarchical data base management systems that will support multiple external views of the data base. Programming of an interface for use with one major DBMS is currently in progress under the direction of one of the authors.

References

1. Dale, A. G. and Dale, N. B., "Schema and Occurrence Structure Transformations in Hierarchical Systems," ACM-SIGMOD International Conference on Management of Data, 1976.

2. Wang, J., Branch Set Preserving Transformations of Hierarchical Data Structures, M. A. Thesis, The University of Texas at Austin, December, 1976.

3. Parsons, R. G., et.al., "Data Manipulation Language Requirements for Data Base Management Systems, The Computer Journal, Vol. 17, No. 2, May, 1974.