A Method for the Logical Design of the Hierarchical Data Model

HIROTAKA SAKAI*

A logical database design generally consists of two stages: designing the conceptual schema, and translating it into object data models acceptable by the underlying database systems. In this paper, the Entity-Relationship model is used to describe the conceptual schema, and the translation process from the Entity-Relationship model to the hierarchical model is proposed. In the translation process, the existence dependencies and the hierarchical decompositions which represent the inherent hierarchical properties of information are analyzed. The query characteristics are also investigated and formulated in the form of the access weight matrix.

1. Introduction

A logical database design is a process of modeling the real world and organizing data in the framework of data models acceptable by the underlying database systems. The process generally consists of two stages: (1) designing the conceptual schema, and (2) translating it into the hierarchical, network, or relational schemas. The conceptual schema is a description of the enterprise view of data and should be designed to reflect the inherent structural properties of information. The hierarchical, network, or relational data model to be used in the actual applications environments should be derived from the stable conceptual schema.

We use the Entity-Relationship model (abbr. ER model) to describe the conceptual schema, and establish a systematic process to derive the hierarchical data structure from the conceptual schema.

The ER model, initially proposed by Chen, has a rich semantic expressiveness and is now widely used in logical database design under the name of the Entity-Relationship approach [1, 2, 3]. The translation process from the ER model to the hierarchical model has recently been studied by several authors [7, 8, 10].

In this article, we propose a method for the data translation based on the following improved ideas.

(1) Analyze the hierarchical properties in the ER model based on the notions of the existence dependency and the hierarchical decomposition which express certain types of semantic constraints, and reflect them in the translation process.

(2) Analyze the characteristics of query requests to the database, and define the access weight to measure the load that queries impose on the database. Reflect this measure in the hierarchical ordering in the target model.

*Hitachi Institute of Technology, Hitachi, Ltd.

2. The ER Model

2.1 The Description of E Sets and R Sets

In the ER model, information is represented using three conceptual elements: entities, relationships among entities, and values. The set of similar entities, similar relationships, and similar values in certain contexts are called an E set, an R set, and a V set respectively [1, 2, 9, 11].

An E set is described in the form E $(A_1, A_2, \cdots, A_n)$, where E is the name of the E set and $A_1, A_2, \cdots, A_n$ are attributes. The attribute A_i is a property of an E set, and is defined as a function from E into the associated V set V_i.

An R set is a set to relate several (not necessarily distinct) E sets $E_1, E_2, \cdots,$ and E_m. It is a set of tuples $(e_1, e_2, \cdots, e_m)$ of mutually related entities e_i of E_i $(i=1, 2, \cdots, m)$.

An R set is described in the form $R(E_1, E_2, \cdots, E_m: A_1, A_2, \cdots, A_n)$, where R is the name of the R set and each E set E_i has a specific role in R. We say R is defined on the E sets $\{E_1, E_2, \cdots, E_m\}$. An R set may have attributes $A_1, A_2, \cdots, A_n$ also. In this case, the attribute A_j is a function from R into the associated V set V_j.

The ER model is constructed as a set of E sets and R sets, and is illustrated in the ER diagram, in which the E set and the R set are represented by rectangular- and diamond-shaped boxes respectively as they appear in the examples.

The ER model can be constructed in the normalized form. The normalization of the ER model is similar to that in the relational model[4] and was discussed in other papers[9, 11].

This notion provides a theoretical foundation for the data segmentation. We assume that every conceptual schema designed in the ER model has been normalized.

2.2 Operations on R Sets

Let E, F, and G be E sets and R(E, F, G) be an R set defined on {E, F, G}. The projection of R over E, denoted R[E], is defined by $R[E]=\{e \in E | \exists f \in F \exists g \in G: (e, f, g) \in R\}$.

The join operation is used to make a connection between E sets that appear in different R sets. Let R(E, F) and S(F, G) be two R sets defined on sets of E sets {E, F} and {F, G} respectively. The join of R and S, denoted R*S, is defined by $R*S=\{(e, f, g) | (e, f) \in R \wedge (f, g) \in S\}$.

The above definitions are valid when each of E, F, and G is a set of E sets. If for example E is a set of E sets, then we simply replace an element of E with a tuple of elements of the constituent E sets of E.

3. Hierarchical Properties in the ER Model

3.1 The Existence Dependency

There exist certain types of hierarchical properties in the ER model. These properties are often found in the semantic constraints concerning R sets.

Among other things, we shall consider the existence dependencies in E sets and R sets and the hierarchical decompositions of R sets, and reflect these structural properties in the translation into the hierarchical data model.

The notion of the existence dependency (abbr. ED) in the ER model provides hierarchical properties in the sense that the existence of elements of an E set or an R set depends on the existence of elements of other or the same E sets or R sets.

We can recognize the following types of existence dependencies.

(1) Existence Dependencies between E Sets

Let E, F be two E sets and R be an R set defined on {E, F}. When any entity f of F exists only if f is related to some entity e of E by a relationship of R, we say F is existentially dependent on E in R, and write ED F→E in R.

An example is the R set SUPPORT which is defined on the E sets {EMPLOYEE, DEPENDENT}. Here we have ED DEPENDENT→EMPLOYEE in SUPPORT. Another example is the R set INSTANCE-OF-FLIGHT which is defined on the E sets {FLIGHT, OPEN-FLIGHT}. An entity of FLIGHT stands for a flight type like 'JL001' while an entity of OPEN-FLIGHT represents an instance of a flight type by date like 'JL001 on December 11'[12]. In this case we have ED OPEN-FLIGHT→FLIGHT in INSTANCE-OF-FLIGHT.

From this type of the existence dependency ED F→E in R, the hierarchy with the parent E and the child F is naturally constructed.

(2) Existence Dependencies between R Sets

Let R(E, F) and S(F, G) be two R sets defined on E sets {E, F} and {F, G} respectively. When any relationship $s=(f, g)$ of S ($f \in F$, $g \in G$) exists only if some relationship $r=(e, f)$ of R ($e \in E$) exists, we say S is existentially dependent on R and write ED S→R. In this situation, we can consider the hierarchical ordering of E sets with the parent E, the child F, and the grandchild G.

Consider for example the E sets FACTORY, PRODUCT, CUSTOMER, and the R sets MAKE (FACTORY, PRODUCT) and USE(CUSTOMER, PRODUCT). If we have ED USE→MAKE, then the hierarchical order the parent FACTORY, the child PRODUCT, and the grandchild CUSTOMER is naturally constructed. If, at the same time, we had to consider the condition ED MAKE→USE, then R set MAKE and USE are mutually existentially dependent. In order to determine the hierarchical order between the E sets in this situation, it is necessary to analyze the behavioral properties on the database.

There is yet another type of the existence dependency where an R set is transitively dependent on other R sets. Let R(E, F), S(F, G), and T(E, G) be R sets defined on E sets {E, F}, {F, G}, and {E, G} respectively. When any relationship $t=(e, g)$ of T ($e \in E$, $g \in G$) exists only if a relationship $r=(e, f)$ of R and a relationship $s=(f, g)$ of S ($f \in F$) exist, then we say T is transitively existentially dependent(abbr. TED) on R and S, and write TED T→(R, S). In this dependency, we have $T \subset (R*S)[E, G]$. Namely, T is structurally a subset of (R*S)[E, G]. In particular the hierarchical property of T depends on the hierarchical properties of R and S.

Consider for example the E sets STUDENT, COURSE, OFFERING, and the R sets ENROLL (STUDENT, COURSE), INSTANCE-OF-COURSE (COURSE, OFFERING), ATTEND(STUDENT, OFFERING) as illustrated in Fig. 1. Here an entity of OFFERING is an instance of an entity of COURSE by date. Therefore we have ED OFFERING→COURSE in INSTANCE-OF-COURSE. We further assume that only the students who have enrolled in the course can attend the offering of that course. This is expressed in TED ATTEND→(ENROLL, INSTANCE-OF-COURSE). The relationship (student, offering) of ATTEND can be represented in a sequence of relationships (student, course) of ENROLL and (course, offering) of INSTANCE-OF-COURSE. If there were a situation where ENROLL could be viewed as a hierarchy with the parent STUDENT and the child COURSE, then ATTEND should also be viewed as a hierarchy with the parent STUDENT, the child COURSE, and the grandchild OFFERING.

3.2 The Hierarchical Decomposition

The notion of the hierarchical decomposition,

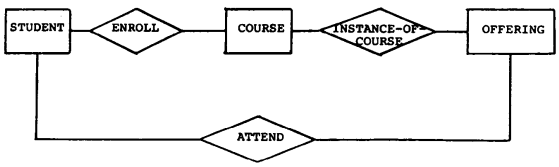

Fig. 1 The ER Diagram including TED.

originally introduced by Delobel in the relational model [6], provides a means to formally analyze the hierarchical properties in the ER model.

Let $R(E_0, E_1, E_2, \cdots, E_m)$ be an R set with no attributes where each E_i $(i=1, 2, \cdots, m)$ is a set of E sets. R is said to obey the first order hierarchical decomposition (abbr. FOHD), if we have $R(E_0, E_1, E_2, \cdots, E_m)=R[E_0, E_1]*R[E_0, E_2]*\cdots*R[E_0, E_m]$, and is denoted FOHD $E_0: E_1|E_2|\cdots|E_m$ in R.

This FOHD means that, for each i and j $(i \neq j)$, all the tuples of entities $\{e_i \in E_i\}$ which are related to a tuple of entities $e_0 \in E_0$ are also related to all the tuples of entities $\{e_j \in E_j\}$ which are related to the same e_0. In the FOHD, the pair $(E_0: E_i)$ is called the branch with the parent segment E_0 and the child segment E_i.

Among the set of branches $(E_0: E_i)$ generated by the FOHD, there may be certain branches which are further decomposable into the branches with parent segments including E_0. For example, if E_1 is a set of E sets $\{F_0, F_1, F_2, \cdots, F_k\}$, and we have the FOHD $E_0, F_0: F_1|F_2|\cdots|F_k$ in $R[E_0, E_1]$, then the branch $(E_0: E_1)$ is decomposed into subbranches $(E_0, F_0: F_1)$, $(E_0, F_0: F_2), \cdots$, and $(E_0, F_0: F_k)$. Some of these subbranches might again be decomposable into subsubbranches with parent segments including E_0 and F_0. By the repetition of such decompositions, a hierarchical organization of E sets is constructed. This sequence of FOHDs (including the case of a single FOHD) is called the hierarchical decomposition (abbr. HD).

Consider for example an R set $R(E_1, E_2, E_3, E_4, E_5)$ defined on E sets $\{E_1, E_2, E_3, E_4, E_5\}$. If we have the HD (1) $E_1: E_2, E_3, E_4|E_5$ in R, and (2) $E_1, E_2: E_3|E_4$ in $R[E_1, E_2, E_3, E_4]$, then the tree illustrated in Fig. 2(1) is constructed. The HD of a given R set is not necessarily unique. For example, the above R set might obey another HD (3) $E_3: E_4|E_1, E_2, E_5$ in R, and (4) $E_3, E_2: E_1|E_5$ in $R[E_3, E_2, E_1, E_5]$ resulting another tree illustrated in Fig. 2(2). The hierarchical representation obeying the HD is discussed in Sec. 5.

4. Query Analysis

4.1 Formalization of Queries

In addition to the analysis of structural properties of data, it is required to reflect behavioral properties of data in the construction of the hierarchical structure. For simplicity, we shall consider only query requests to the database. In order to analyze characteristics of queries, we formalize the queries to access data elements in the ER model. We first define a simple query in which a reference to an R set is made at most once. Any query is then represented as an appropriate sequence of simple queries.

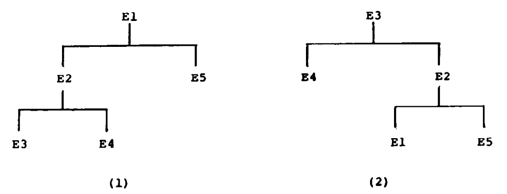

Fig. 2 The Tree Representations of the HDs.

Let R be an R set defined on E sets $\{E_1, E_2, \cdots, E_m\}$, a simple query Q which refers to R is described in the form $Q(R)=\langle C: T\rangle$ where C is the search criteria and T is the target list of Q.

The search criteria C is to specify the condition which should be satisfied by the data elements accessed, and might be expressed in the form of Boolean expressions including attributes of $E_1, E_2, \cdots, E_m$, or R, and values of their associated V sets. However, at this stage of the design, it is sufficient to know only which of $E_1, E_2, \cdots, E_m$ and R participate in the search criteria. The target list T is the set of entities or relationships of $E_1, E_2, \cdots, E_m$, or R, or subtuples of them which are the data objects to be accessed. Here again it is sufficient to consider what E sets or R set of $E_1, E_2, \cdots, E_m$, and R paticipate in the target list. Therefore both C and T are denoted in the forms of subsets of $\{E_1, E_2, \cdots, E_m, R\}$.

Consider for example the E sets SUPPLIER (SNO, SNAME, SCITY) and PART (PNO, PNAME, PCOLOR) and the R set SP (SUPPLIER, PART: QTY). When we have two queries $Q_1(SP)$ and $Q_2(SP)$ stating "Get SNO for SUPPLIER who supply PART P2" and "Get SNAME for SUPPLIER who supply at least one red PART" respectively, then each of $Q_1(SP)$ and $Q_2(SP)$ is described in the same form $\langle$PART: SUPPLIER$\rangle$. A query which does not refer to any R set is treated as a special form of a simple query. For example the query "Get PNAME for red PART" is denoted $Q(\phi)=\langle$PART: PART$\rangle$.

Any query can be decomposed into a sequence of simple queries. Namely, the query referring to R sets $R_1, R_2, \cdots$, and R_k is decomposed into a sequence of $Q(R_1), Q(R_2), \cdots$, and $Q(R_k)$.

The R sets which are referred to in the query are not necessarily distinct. Consider for example a query "Get SNAME for SUPPLIER who supply at least one PART supplied by SUPPLIER S_2." In this query, the R set SP is referred to twice, first for retrieving the PART supplied by S_2 and secondly for accessing the SUPPLIER who supply the PART retrieved. The query is therefore decomposed into a sequence $Q_1(SP)=\langle$SUPPLIER: PART$\rangle$ and $Q_2(SP)=\langle$PART: SUPPLIER$\rangle$.

4.2 The Access Weight Matrix

The access weight of a query on an E set is introduced to measure the load put on an E set by a query. This measure reflect the usage characteristics of the ER model and should be analyzed in the data translation process.

Let R be an R set defined on E sets $\{E_1, E_2, \cdots, E_m\}$ and $\{Q_i(R)=\langle C_i: T_i\rangle\}$ be a set of queries which refer to R. If E_j appears in C_i or T_i, the access weight of $Q_i(R)$ on E_j depends on two factors: the weight of $Q_i(R)$ itself, denoted $w(Q_i(R))$, and the degree of reference of E_j in $Q_i(R)$, denoted $r(E_j, Q_i(R))$, which reflects the way the E set E_j is referred to in $Q_i(R)$.

The weight of $Q_i(R)$ is defined by $w(Q_i(R)) = t \times f$. In this expression, t is the responsiveness parameter taking a relatively high value for on-line queries, and f is the access frequency of $Q_i(R)$ executed per unit time.

The degree of reference of E_j in $Q_i(R) = \langle C_i : T_i \rangle$ is defined as follows.

$$r(E_j, Q_i(R)) = \begin{cases} s \text{ if } E_j \in C_i (s > 1) \\ 1 \text{ otherwise (i.e. } E_j \in T_i - C_i) \end{cases}$$

Namely, r takes a higher value when E_j is accessed in the search criteria.

Now the access weight of $Q_i(R)$ on E_j, denoted $w(E_j, Q_i(R))$, is given by the following expression.

$$w(E_j, Q_i(R)) = w(Q_i(R)) \times r(E_j, Q_i(R)).$$

If we consider $Q(R) = \{Q_i(R)\}$, the set of all simple queries referring to R, then the access weight of Q(R) on E_j, denoted $W(E_j, Q(R))$ is defined as the sum of the quantities $\{w(E_j, Q_i(R))\}$ for all the $Q_i(R)$ of Q(R). Especially, for $Q(\phi) = \{Q_i(\phi)\}$, the set of all simple queries not referring to any R set, $W(E_j, Q(\phi))$ can also be defined in the same way.

Finally, overall status of the access weights imposed on a given ER model is represented by the access weight matrix.

Let $M = \{E_1, E_2, \cdots, E_h, R_1, R_2, \cdots, R_k\}$ be the ER model, and $Q = \{Q(\phi), Q(R_1), Q(R_2), \cdots, Q(R_k)\}$ be the set of simple queries to be executed. The access weight matrix AW of Q on M is then defined as follows.

$$AW = \begin{Bmatrix} W(E_1, Q(\phi)), & W(E_2, Q(\phi)), & \cdots, & W(E_h, Q(\phi)) \\ W(E_1, Q(R_1)), & W(E_2, Q(R_2)), & \cdots, & W(E_h, Q(R_2)) \\ \cdots\cdots & \cdots\cdots & \cdots & \cdots\cdots \\ W(E_1, Q(R_k)), & W(E_2, Q(R_k)), & \cdots, & W(E_h, Q(R_k)) \end{Bmatrix}$$

In the above expression, if the E set E_j is not included in the R set R_i, the value zero is assigned to $W(E_j, Q(R_i))$.

5. Data Translation from the ER Model to the Hierachical Model

5.1 Outline

Let us assume that we have a normalized ER model, and the access weight matrix of queries on the ER model is known. A hierarchical data structure is constructed by applying a certain translation process to the ER model.

The process is established based on the following principles.

(1) The inherent hierarchical properties that exist in the ER model should be inherited to the hierarchical model after the translation.

(2) If there is no structural constraint, the access weight of the queries on each E set should be reflected upon hierarchical ordering of E sets. Especially, the E set with a heavy access weight should be qualified as a root segment of a tree.

The translation process consists of two phases: the local translation and the global translation. The local translation is performed for each R set. The E sets included in the R set are transformed into a hierarchical path. In this process, the structural and behavioral properties of data are analyzed to make proper hierarchical paths. The global translation is a process to concatenate hierarchical paths and make up a final hierarchical structure as an integrated view of data. The E sets with heavy access weights are specified to be the root segments of trees. As a result of this process, an integrated hierarchical structure consisting of a single tree or a set of separate trees is obtained.

5.2 The Local Translation

5.2.1 Hierarchical Paths

In Sec. 3, we have used the notation (A: B) for a branch with a parent segment A and a child segment B. The branch with a parent A and children $B_1, B_2, \cdots, B_n$ is denoted $(A: B_1, B_2, \cdots, B_n)$. A tree is a single branch or is constructed by concatenating branches. We consider the tree in Fig. 2(1) is constructed by concatenating two branches $(E_1: E_2, E_5)$ and $(E_2: E_3, E_4)$, and denote it $(E_1: (E_2: E_3, E_4), E_5)$. In this tree, the segment E_1 is called the root segment. In our discussion, each of the segment $E_1, E_2, E_3, \cdots$ is either an E set or a set of E sets.

We shall first define a general rule to generate a hierarchical path associated with an R set. Let R be an R set defined on E sets $\{E_1, E_2, \cdots, E_{m-1}, E_m\}$. Let us assume that for the access weights $W(E_i, Q(R))$ $(i = 1, 2, \cdots, m)$, the condition $W(E_1, Q(R)) \geqq W(E_2, Q(R)) \geqq \cdots \geqq W(E_m, Q(R))$ holds. The hierarchical path associated with R, denoted HP(R), is the tree $(E_1: (E_2: (\cdots(E_{m-1}: E_m))))$. If a set of attributes X is included in R, then the leaf segment E_m is replaced with the new E set which is generated by adding X to the attributes of E_m. If, for example, we have an R set R(PART, PROJECT, SUPPLIER: QTY) with the access weights W (PART, Q(R)) ≧ W (PROJECT, Q(R)) ≧ W (SUPPLIER, Q(R)), then HP(R) is the tree (PART: (PROJECT: SUPPLIERX)) where SUPPLIERX is the E set generated by adding QTY to the set of attributes of SUPPLIER.

5.2.2 Hierarchical Paths for EDs and HDs

When there exist hierarchical properties such as EDs and HDs in the ER model, then the hierarchical path is constructed according to different rules which precede the general rules mentioned above.

(1) Hierarchical Paths for EDs

If there exist EDs, it is natural that the hierarchical paths should be defined in the following ways according to the types of EDs.

(a) ED between E sets: If "ED F→E in R(E, F)" holds, then we have (E: F) for HP(R).

(b) ED between R sets: If "ED $S(F, G) \rightarrow R(E, F)$" holds, then we have $(E:(F:G))$ for the concatenation of HP(R) and HP(S).

(c) TED between R sets: If "TED $T(E, G) \rightarrow (R(E, F), S(F, G))$" holds, and if the concatenation of HP(R) and HP(S) is representable in $(E:(F:G))$, then we have also $(E:(F:G))$ for HP(T) which is a subset of the concatenation of HP(R) and HP(S) in occurence.

In (b) and (c), the rules for the concatenation of hierarchical paths which precede the general rules of the global translation are also included.

(2) Hierarchical Paths for HDs

Let $R(E_0, E_1, E_2, \cdots, E_m)$ be an R set obeying the HD, then HP(R) is defined as the tree which is obtained from the sequence of FOHDs. Suppose we have the FOHD $E_0: E_1|E_2|\cdots|E_m$ in R, then the tree $(E_0: E_1, E_2, \cdots, E_m)$ is constructed. Suppose further, for example, we have FOHD $E_0, F_0: F_1|F_2|\cdots|F_k$ in $R[E_0, E_1]=R[E_0, F_0, F_1, F_2, \cdots, F_k]$, then the tree $(E_0: (F_0: F_1, F_2, \cdots, F_k), E_2, \cdots, E_m)$ is obtained by replacing E_1 with the branch $(F_0: F_1, F_2, \cdots, F_k)$. This process is repeated until there is no more FOHD in the sequence. When the sequence of the ramifications corresponding to the FOHDs terminates, we have the final tree as the hierarchical path associated with R. However, if there are several different HDs for the same R, then the most desirable HD should be chosen based on the access weight analysis. Accordingly, at each stage of the FOHD and the corresponding ramification of the tree, the branch should be chosen so that the E sets with the higher access weights are included in its parent segments.

5.3 The Global Translation

The hierarchical paths generated in the local translation are concatenated together and integrated into a hierarchical structure.

Two trees including the common segment are cocatenated into a single tree according to the following rules.

(1) The tree (A: B) and (A: C) with a common root segment A are concatenated into a single tree (A: B, C).

(2) The tree (A: B) and (B: C), where the common segment B is a root of one tree and a non-root of another, are concatenated into a tree (A: (B: C)), if the set of occurrences of B in (A: B) includes the set of occurrences of B in (B: C). If it is possible to concatenate (B: C) to more than one tree with the non-root segment B, then (B: C) should be concatenated to the tree in which B is the highest level (i.e. nearest the root segment).

In addition to these rules, it is necessary to verify the semantic integrity of the concatenation process. However, this problem is beyond the scope of this article and is left to the decision of the designers.

In the global translation, it is necessary to pay particular attention to dealing with the hierarchical path with a heavy access weight. An E set E which is the root segment of a hierarchical path associated with an R set R, is said the absolute root segment of HP(R), if the access weight $W(E, Q(R))$ is larger than a specified large number M. Then the rules of the concatenation is supplemented by the following statement.

(3) The hierarchical path with an absolute root segment should not be concatenated to any hierarchical paths except for those with the same root segment.

5.4 Example

We shall give a brief example of the data translation process. Suppose we have a normalized ER model illustrated in Fig. 3.

Let us assume that all the R sets have no attribute and that we have the following EDs, HDs, and the access weight matrix. As for the name of the R sets, we shall use the symbolic names $R_1, R_2, R_3, \cdots$, instead of the semantic names CHARGE, BELONG, ASSIGNED-TO, $\cdots$

(i) ED OFFERING→COURSE in R_5

(ii) TED $R_6 \rightarrow (R_4, R_5)$

(iii) FOHD COURSE: TEACHER|TEXT in R_3

(iv) Access Weight Matrix

	CLASS	TEACH-ER	STU-DENT	TEXT	COU-RSE	OFFER-ING
ϕ	—	—	1000	—	1000	—
R_1	10	100	—	—	—	—
R_2	100	—	1000	—	—	—
R_3	—	1000	—	10	5000	—
R_4	—	—	10000	—	5000	—
R_5	—	—	—	—	10000	1000
R_6	—	—	5000	—	—	1000

Then the translation to the hierarchical model is performed in the following way.

a. Local Translation

We have $HP(R_5)=(COURSE: OFFERING)$ from

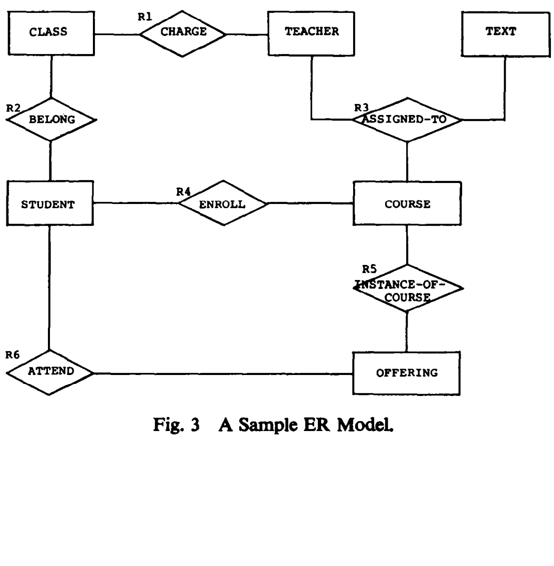

Fig. 3 A Sample ER Model.

(i), and $HP(R_3)$=(COURSE: TEACHER, TEXT) from (iii). The TED (ii) suggests that $HP(R_6)$ should be represented as a part of the concatenation of $HP(R_4)$ and $HP(R_5)$. Analyzing the access weights W(TEACHER, $Q(R_1)$), W(CLASS, $Q(R_1)$), W(STUDENT, $Q(R_2)$), W(CLASS, $Q(R_2)$), W(STUDENT, $Q(R_4)$), and W(COURSE, $Q(R_4)$), We have $HP(R_1)$=(TEACHER: CLASS), $HP(R_2)$=(STUDENT: CLASS), and HP (R_4)=(STUDENT: COURSE) respectively. After the completion of the local translation, we have five hierarchical paths $HP(R_1)$, $HP(R_2)$, $HP(R_3)$, $HP(R_4)$, and $HP(R_5)$ as illustrated in Fig. 4. Note that the R set R_6 does not explicitly appear in the hierarchical path, because we consider $HP(R_6)$ to be embedded in $HP(R_4)$ and $HP(R_5)$.

b. The Global Translation

Let us assume that the set of TEACHER in $HP(R_1)$ is included in the set of TEACHER in $HP(R_3)$, and that the E sets STUDENT in $HP(R_4)$ and COURSE in $HP(R_5)$ are specified as the absolute root segments. Then $HP(R_5)$, $HP(R_3)$, and $HP(R_1)$ are concatenated into the tree (COURSE, OFFERING, (TEACHER: CLASS), TEXT). $HP(R_4)$ and $HP(R_2)$ are also concatenated. At this stage, $HP(R_6)$ which did not appear

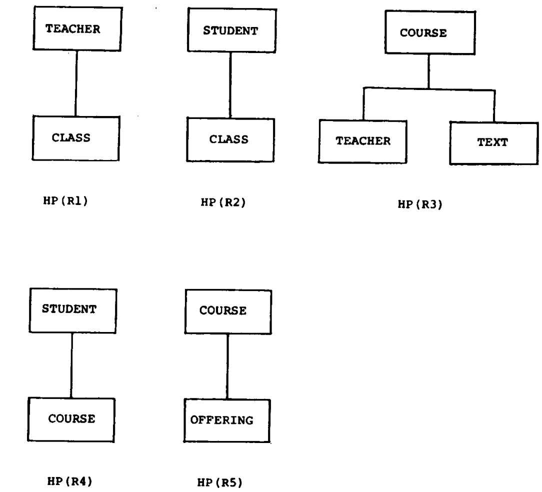

Fig. 4 The Hierarchical Paths constructed by the Local Translation.

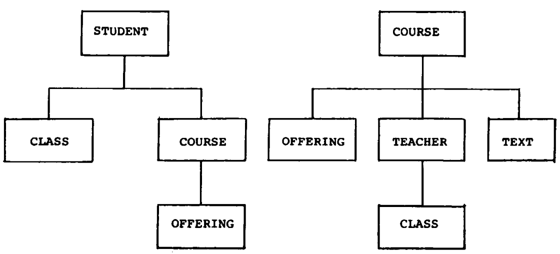

Fig. 5 The Hierarchical Model constructed by the Global Translation.

in Fig. 4, should be represented in the form (STUDENT: (COURSE: OFFERING)) and added to the concatenation. Thus we have another tree (STUDENT: (COURSE: OFFERING), CLASS). After all, we have the hierarchical structure consisting of two trees illustrated in Fig. 5. Note that the E sets STUDENT and COURSE stay at the root segment positions.

6. Data Translation from the ER Model to the IMS Data Model

6.1 The IMS Data Model

The translation rules discussed in Sec. 5 can be applied, with a slight modification, to the data translation from the ER model to the IMS data model. Besides the ordinary hierarchical data structure, IMS supports a restricted form of the network data structure which is represented by the use of the logical relationship[5].

Without the loss of generality, we shall consider here only the bidirectional logical relationship with virtual pairing. Basic patterns of data structures using the logical relationship are illustrated in Fig. 6. In this figure, the symbol * denotes the pointer segment and the dotted line denotes the logical relationship. The branch (A: *) and (B: *) represent the physical and logical parent/child relationships respectively. In the case that B is identical with A as illustrated in Fig. 6(2), the branch (A: *) represents both of the physical and logical relationships.

6.2 The Modified Local Translation

When the hierarchical path associated with each R set has been constructed using the local translation rules described in 5.2, we can further modify it so that the segments with heavy access weights are put in the higher level in the hierarchy.

Let HP(R) be the hierarchical path associated with an R set R, and B be the non-root segment in HP(R) with the access weight W(B, Q(R)) which is larger than a specific value M. We can replace the segment B with a pointer segment * and construct a new branch (B: *) representing the logical parent/child relationship as

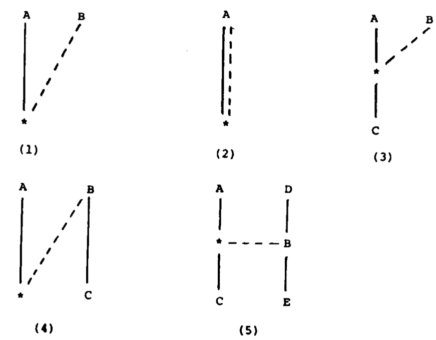

Fig. 6 Data Structures including the Logical Relationship.

illustrated in Fig. 6(1).

6.3 The Modified Global Translation

In order to construct the final IMS data model, the global translation rules (1), (2), and (3) in 5.3 are modified by supplementing the following (1′), (2′), and (3′) respectively, and applied to the set of the hierarchical paths obtained in the modified local translation rules.

(1′) In rule (1), the common root segment A is allowed to be a logical parent.

(2′) In rule (2), the common segment B is allowed to be a logical child in (A: B) or a logical parent in (B: C).

(3′) Let (A: B) and (B: C) be trees where the segment B is qualified as an absolute root segment in (B: C). Then the trees (A: B) and (B: C) can be concatenated into the structure with the logical parent B as illustrated in Fig. 6(4).

However, the use of the logical relationship is restricted by the IMS rules which are dependent upon the implementation of the system[5]. It is therefore necessary to verify the legality of the concatenation in the modified global translation.

Applying the modified translation rules to the ER model in the example in 5.4, the IMS data model illustrated in Fig. 7 is constructed.

7. Conclusion

The data translation process from the ER model to the hierarchical model is summarized as follows.

(1) Normalization: It is first assumed that the given ER model has been normalized.

(2) Query Analysis: All the queries are analyzed and formulated in the form of the access weight matrix.

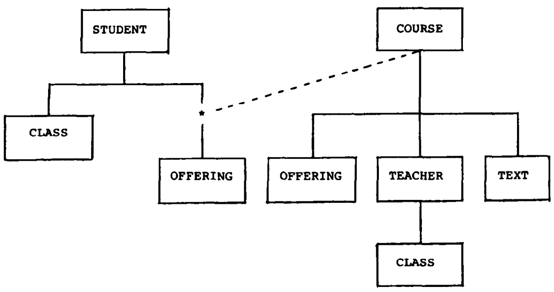

Fig. 7 The IMS Data Model constructed by the Modified Translation.

(3) Local Translation: The hierarchical path associated with each R set is organized based on the analysis of the inherent hierarchical properties such as EDs and HDs in the ER model and the access weights.

(4) Global Translation: The final hierarchical model is constructed by concatenating the hierarchical paths. The E sets with heavy access weights are allowed to stay as the absolute root segments of the trees.

The translation process mentioned so far provides a unified approach to the logical design of the hierarchical model. However, different hierarchical models might be generated depending on the specifications of the access weights or the absolute root segments. Some methods or tools for evaluating the hierarchical models are required.

References

1. Chen, P. P. The Entity-Relationship Model: Towards a Unified View of Data, *Transactions on Database Systems* 1, 1, (1976), 9–36.
2. Chen, P. P. The Entity-Relationship Model: A Basis for the Enterprise View of Data, *AFIPS Conf. Proc.* 46, AFIPS Press. (1977), 77–84.
3. Entity-Relationship Approach to Systems Analysis and Design, Chen, P. P. Ed. (North-Holland, Amsterdam, 1980).
4. Codd, E. F. Further Normalization on the Database Relational Model, *Courant Computer Science Symposium* 6, Database Systems (Prentice-Hall, Englewood Cliffs, 1971), 65–98.
5. Date, C. J. An Introduction to Database Systems (2nd edition) (Addison Wesley, Reading, MA, 1977).
6. Delobel, C. Normalization and Hierarchical Dependencies in the Relational Data Model, *ACM Transactions on Database Systems*,3 , 3, (1978), 201–222.
7. Lusk, E. L., Overbeek, R. A. and Parrello, B. A Practical Design Methodology for the Implementation of IMS Databases, Using the Entity-Relationship Model, *Proc. ACM-SIGMOD* 1980, (1980) 9–21.
8. Ng, P. A. Further Analysis of the Entity-Relationship Approach to Database Design, *IEEE Transactions on Software Engineering*, SE-7, 1, (1981), 85–99.
9. Sakai, H. On the Optimization of the Entity-Relationship Model, *Proc. 3rd USA-JAPAN Computer Conf.* (1978), 145–149.
10. Sakai, H. A Unified Approach to the Logical Design of a Hierarchical Data Model, in: Entity-Relationship Approach to Systems Analysis and Design, (North-Holland, Amsterdam, 1980), 61–74.
11. Sakai, H. Entity-Relationship Approach to the Conceptual Schema Design, *Proc. ACM-SIGMOD* 1980, (1980), 1–8.
12. Sakai, H. A Method for Defining Information Structures and Transactions in Conceptual Schema Design, *Proc. 7th Int. Conf.* on Very Large Data Bases (1981), 225-234.
13. Smith, J.M. and Smith, D.C. Database Abstractions:Aggregation and Generalization, *Transactions on Database Systems* 2, 2, (1977), 105–133.

(Received June 17, 1981)