Realizing System Entity Structure in A Relational Database

Tag Gon Kim
Department of Electrical and Computer Engineering
Unviersity of Kansas
Lawrence, KS 66045

ABSTRACT

System entity structure (SES) developed by Zeigler is a structural knowledge representation scheme that contains knowledge of decomposition, taxonomy, and coupling of a system. Formally, SES is represented by a set of entities and three relationships defined on entities in the set. This paper describes a realization of SES in a relational database system. The benefit of such a realization is that the system entity structure provides the semantics for representation of systems structure while the relational database serves as an implementation language. An example for constructing a system entity structure, using a pseudo-SQL, for a computer-based control system is given.

1. INTRODUCTION

In large scale modeling and simulation for systems design, the modeler often employs a multifaceted modeling methodology that deals with the multiplicity of facets for a real world system in a coherent manner. Within the modeling methodology, the modeler represents a family of possible alternative configurations of a system to be modeled in an organized form. As design objectives and requirements are clarified, the modeler selects a combination of components as a design candidate from the family of configurations and evaluate it with respect to design objectives (Rozenblit et. al., 1990).

The system entity structure formalism developed by Zeigler (Zeigler, 1984) is a representation scheme for systems structure that has been used as a formalism for the multifaceted modeling. It contains knowledge of decomposition, taxonomy, and coupling of a system. The operation *prune* defined on the system entity structure serves as a basis to select a candidate out of possible alternatives for evaluation. By combining a simulation modeling environment such as DEVS-Scheme (Kim and Zeigler, 1990), the system entity structure serves as a model base management system for simulation models in systems design (Kim et. al., 1990). In fact, a realization of such a model base management system called ESP-Scheme (Kim and Zeigler, 1989) is in place in which a LISP dialect called Scheme is employed as an implementation language.

This paper describes a realization the system entity structure in a relational database system. The benefit of such a realization is that the system entity structure provides the semantics for representation of systems structure while the relational database serves as an implementation language. Thus, the realization is a model base management system in a relational database that can be used as a multifaceted modeling and simulation tool for complex systems design.

2. SYSTEM ENTITY STRUCTURE FORMALISM

A system entity structure (SES) is a representation scheme for the structure of a system. It has been graphically represented as a labeled tree with attached variable types that satisfies five axioms – alternating mode, uniformity, strict hierarchy, valid brothers, and attached variables. Detail description of the axioms is beyond of the scope of this paper and is available in (Zeigler, 1984).

Within the system entity structure formalism, a real world system can be represented by a set of entities and three relationships between entities. An entity within the entities set represents a component of the real system. Each such entity has one or more aspect and/or specialization. An aspect is a decomposition relationship between an entity and its subentities in which the subentites represent subcompoments of a component represented by the entity. A specialization is a taxonomic relationship between a general entity and specialized ones. Associated with an aspect is a coupling relationship between a pair of entities with a designated ports pair. There is a special aspect relationship called a multiple aspect. A multiple aspect is an one-to-many relationships between an entity and its homogeneous subentities. Thus, a component represented by an entity having a multiple aspect consists of a collection of homogeneous components. We call such an entity a multiple entity.

3. RELATION AND RELATIONAL DATABASE

We briefly discuss concepts of mathematical relation on which a relational database system is based. The concepts discussed here are used in discussion of compatibility between relation and the system entity structure formalism.

3.1. Relation

The term "relation" is essentially a mathematical term of a table. Formally, a relation R on sets D_1, D_2,...., D_n is defined as:

$$R = \{ (v_1, v_2, ..., v_n) \mid v_1 \in D_1, v_2 \in D_2, ..., v_n \in D_n \}$$

where, the sets D_1, D_2,..,D_n are called domains of attributes, and $(v_1, v_2, ..., v_n)$ is called a tuple. Since a relation is a mathematical set, the order of tuples and attributes in relations has no significance. From now on, we use relation and table interchangeably.

As mathematical sets, relations possess the following properties (Date, 1990):

1. Tuples are not duplicated.
2. Tuples are unordered.
3. Attributes are unordered.
4. All attribute values are atomic.

The properties 1, 2, and 3 follow from the properties of a mathematical set. The property 1 implies that there always exists a primary key in a relation since tuples are unique. To satisfy the property 4, the value at every (row, column) position within the table can not be a list. A relation satisfying this condition is said to be normalized.

3.2. Relational Database

A relational database is a database that is perceived by its users as a collection of tables. More precisely, tables in the relational database are time-varying normalized relations of assorted degrees (Date, 1990). In the relational database, information about the real world can be represented by a collection of entities relations and a collection of relationships relations on the entities. It is known that there is nothing that can be represented by a hierarchy that cannot be represented by relations alone. More importantly, organizing information about the real world using relations is much simpler than using a hierarchy.

The simplicity follows from the fact that relations do not employ links connecting one information to another and that there are no ordering in relations. The links in the hierarchical structure can be represented by the relationships in the relational structure. In fact, the hierarchical structure fundamentally requires certain additional data access operators which are not required for the equivalent relational structure. Thus, the links of the hierarchical structure certainly do not add any power but serve only to add complexity. Ordering in operations for constructing and/or destructing a hierarchy also serves to add complexity in the hierarchical system.

4. SES AS SEMANTICS FOR SYSTEM STRUCTURE

We now examine how a system entity structure can be represented by a relational database. Based on the discussion of the system entity structure and the relation/relational database, we see that a system entity structure can be realized in a relational database. In fact, semantics of the system entity structure seems to be highly compatible to the underlying relational model of the relational database. Note that the system entity structure bears no relation to the Entity-Relationship Data Model proposed by Chen (Chen, 1976). However, attempts for using the system entity structure in database design have been made (Higa, 1988) and the results of such attempts have shown to be superior to conventional schemes.

Let us precisely examine the compatibility between the system entity structure and the relational model. There exists a correspondence between representation of a system using the system entity structure and representation of the real world using relations. As we discussed earlier, a system entity structure can be represented by a set of entities and three relationship sets, namely, an aspects relationship set, a specializations relationship set, and a couplings relationship set. Since the entities within the set represent components in a system, the set can be represented by an entities relation in a relational system. The aspects set, each element of which represents a relationship between a component and its subcomponents in a system, can be represented by an aspects relationship relation in a relational system. Similarly, the specializations and couplings sets can be represented by an specializations relationship relation and an couplings relationship relation, respectively. Thus, a system entity structure can be represented by a set of the four relations described above.

5. REALIZING SES IN RELATIONAL DATABASE

To realize a system entity structure in a relational database, we first give relational data representation for the system entity structure and then define operations on such relations. As explained earlier, relational data representation is to define four relational tables for a system entity structure. Operations on such relations include create/destroy tables, update tables, and queries on tables.

5.1. Relational Data Representation

A table in a relational system consists of a row of column headings and zero or more rows of data values. A system entity structure in a relational database consists of four relations: the entities relation, the aspects relationship (ASPRS) relation, the specializations relationship (SPECRS) relation, and the couplings relationship (COUPRS) relation.

Formally, a system entity structure (SES) in a relational system is defined as:

$$SES = < ER, ASPR, SPECR, COUPR >$$

where ER : ENT relation
ASPR : ASPRS relation
SPECR : SPECRS relation
COUPR : COUPRS relation
with constraints:

$$ASPRS = \{ (e_i, e_j) \mid e_i, e_j \in ENTSET \}$$
$$SPECRS = \{ (e_k, e_l) \mid e_k, e_l \in ENTSET \}$$
$$COUPRS = \{ (e_m, e_n) \mid e_m, e_n \in ENTSET \}$$

For each relation defined above, we define column headings with asscociated data types and a primary key. The entities relation consists of three column headings, namely, EntNumber (entity id number), EntName (entity name), and IsComponent (is this entity a component of another entity?). Data types of EntNumbe, EntName, and IsComponent are string, string, and boolean, respectively. Since EntName may not be unique from the axiom for the system entity structure, we use EntNumber as a primary key for the entities relation.

We define three column headings for the aspects relationship relation as AspName (aspect name), EntName

(entity which has this aspect), and CompEntName (entity which is under this aspect). Data type of AspName is string and the primary key of the aspects relation is (AspName, CompEntName).

The column headings for the specializations relationship relation will be SpecName (specialization name), EntName (entity which has this specialization), and SpecEntName (entity which is under this specialization). SpecName is of type string and (SpecName, SpecEntName) is used as the primary key of the specializations relationship relation.

Finally, the couplings relationship relation has four column headings. They are AspName (aspect which has this couplings relationship), FromEntName (entity name for coupling source), ToEntName (entity name for coupling destination), FromPortName (port name associated with FromEntName), and ToPortName (port name associated with ToEntName). All headings are of type string and the primary key for the couplings relationship relation is (FromEntName, FromPortName).

5.2. Relational Operations

To define four relations for a system entity structure discussed above, there should be means to construct and destruct the corresponding four tables. For this, we define constructor CREATE and destructor DESTROY. The operation CREATE creates a table with the name of table, the column headings, and the primary key. The operation DESTORY destroies existing tables.

To manipulate created tables, we define three basic operations, namely, UPDATE, DELETE, and INSERT and one query SELECT. While the query does not change the contents of tables, the three operations change the state of tables. The operations UPDATE and DELETE modifies and deletes all records in a table, respectively. The operation INSERT inserts records into a table having the specified values for the specified fields. Finally, the query SELECT on a table retrieves specified fields from the table that satisfy the specified condition. It should be noted that the result of the query SELECT forms another relation. The query SELECT can be used to express various kinds of join operations in a relational database.

5.3. Data Consistency in Operations

We now discuss the maintenance of data consistency following the three operations defined above. It is known as a difficult problem to find a set of rules to maintain such data consistency in a relational database. The difficulty follows from that fact that the semantics and consequences of operations, such as deleting and updating, are not clearly defined. We shall show that the system entity structure formalism provides the sounder semantics for the maintenance of data consistency.

The consequences of the operations UPDATE, DELETE, and INSERT on the entities relation may require other operations on the entities and/or relationships relations. The reason is that there may be a relationship between one entity on which such operations are applied and another. For example, deleting a tuple in the entities relation may require deleting some other tuples in the relation. At the same time, certain tuples in the relationships relations whose entity is related to the deleted entity need to be deleted. In general, the consequences of such operations depend on types of operations and kinds of relations on which the operation is applied. Here we shall discuss consequences of each operation defined on the relations representing a system entity structure.

The operation UPDATE, if applied to entities on the entities relation, results in the operations that should be applied on the relationships relations. There are two different cases in the resulting operations. To be specific, if the updated value is not part of an entity primary key, no consequence results in. However, if the value is part of an entity primary key, changes of the entity primary keys in all related relationship relations, namely, aspects, specializations, and couplings relations, would be required.

Deleting an entity tuple using the operation DELETE results in deleting any entity tuple whose entity is in the aspect relationship with the deleted entity. At the same time, the relationship tuples in the relationships relations associated with the deleted entity should be deleted. This procedure needs to be applied recursively.

5.4. Pseduo-SQL for Expressions

We employ SQL (Structured Query Language)-alike formats to express the operations described above. We call such formats the PSQLES (Pseudo SQL for Entity Structure). Expressions in the PSQLES can be mechanically translated into a SQL available in a relational database system. Table 1 shows formats for all operations including constructor and destructor. In the following PSQLES formats, we use the capital letters as key words and the [ ] as optional. The formats below will be used in the example to be given in section 6.2.

Table 1. PSQLES Formats for Expressions

```
CREATE TABLE table-name
{ column-name : data-type [; column-name : data-type ].... ;
PRIMARY KEY : = primary-key };

DESTROY TABLE table-name [; table-name];

SELECT column-name [; column-name]
FROM TABLE table-name
[WHERE condition];

DELETE ROW
FROM TABLE table-name
WHERE condition;

INSERT [ { column-name [; column-name]... } ]
INTO TABLE table-name
WITH VALUES { literal [; literal] ....};

UPDATE TABLE table-name
WITH column-name : = expression [; column-name : = expression]...
[WHERE condition];
```

5.5. Pruning SES

As we mentioned earlier, the operation *prune* on a SES is a procedure to select a candidate design out of many possible alternatives for evaluation with respect to design objectives. Fig. 1 outlines the pruning operation for a SES realized in a relational database.

```
Prune (SES = < ER, ASPR, SPECR, COUPR >)
begin
  create four relations for pruned entity structure PES
  let PES = < ER', ASPR', SPECR', COUPR' >
  if ER has one tuple then
    select the entity tuple and insert it in ER'
    return PES
  else {
    select a tuple of an entity, say ENT, from ER
    insert the tuple in ER'
    ENTSET := { ENT }
    while ENTSET is not empty
      call the first element of ENTSET ENTi
      select a tuple of one aspect, say ASPj, under ENTi
        from ASPR
      insert the tuple in ASPR'
      select all tuples of entities from ER that are re-
        lated to ENTi on ASPj relationship
      select all tuples of couplings from COUPR as-
        sociated with ASPj
      insert the tuples of entities in ER'
      insert the tuples of couplings in COUPR'
      ENTSET := ( ENTSET - {ENTi} ) U { all en-
        tities related to ENTi on ASPj }
      call ENTi current entity
      while current entity has a specialization
        select a tuple of an entity, say ENTj, from
          SPECR that is related to the current entity on
          the specialization
        rename ENTi to ENTj in all tuples having ENTi
          in ER'
        rename ENTi to ENTj in all tuples having ENTi
          in COUPR'
        call ENTj current entity
        end while current entity
      end while ENTSET
    return PES }
end Prune
```

Fig. 1. Pruning SES in Relational Database.

6. STEPS FOR BUILDING SES AND EXAMPLE

We present procedures for building a SES for a system. We apply the procedure to constructing a SES for a computer-based control system the structural specification of which is described informally.

6.1. Steps for Building SES

We realize a system entity structure in a relational database system in the following six steps.

1. identify the entities set.
2. identify the relationship sets defined on the entities set.
   2.1. identify the decompositions relationship set.
   2.2. identify the taxonomies relationship set.
   2.3. identify the couplings relationship set.
3. identify semantic information for the relationship sets.
   3.1. identify subentities for an entity for each decomposition in the decompositions relationship set.
   3.2. identify specialized entities for a general one for each specialization in the taxonomies relationship set.
   3.3. identify a pair of entities for each coupling in the couplings relationship set.
4. define the attributes and their domains.
5. decide primary keys for the entity relation and relationship relations.
6. realize the entity and relationship sets into relations.
   6.1. the entities set into the entities relation.
   6.2. the decompositions relationship set into the aspects relation.
   6.3. the taxonomies relationship set into the specializations relation.
   6.4. the couplings relationship set into the couplings relation.

We now present an example of constructing a SES in a relational database based on the above procedure.

6.2. An Example

Consider the following informal description of system structure for a computerized, real-time control system called a CNTL. We intentionally do not include functional specification of each component in the CNTL.

> "The CNTL consists of a PLANT, a SENSOR, a POWER_AMP, and a CONTROLLER. The PLANT to be controlled can be either a MOTOR or a AIRCON. Therefore, the SENSOR may be either a THERMOMETER or a SPEEDMETER. A PERSONAL COMPUTER (PC), a A/D_CONVERTER, and a D/A_CONVERTER constitute the CONTROLLER. Coupling scheme of the CNTL is as follows. The PLANT has a connection to the outside world through the CNTL. The PLANT's output port "current_value" is connected to the input of the SENSOR "value_in". The SENSOR transmits the value through its output port "measured_out" to the CONTROLLER's input port "in". The CONTROLLER sends the received value to the AD_CONVERTER through its input port "analogy_in." The output port of the AD_CONVERTER "digital_out" is connected to the PC's input port "digital_in." The PC's ouput port "digital_out" is connected to the DA_CONVERTER's input port "digital_in." The DA_CONVERTER's output port "anal-

ogy_out" is connected to the CONTROLLER's output port "out", which then is connected to the POWER_AMP's input port "analogy_in." The POWER_AMP's output port "power_out" is connected to the input port "power_in" of the PLANT"

Based on the above informal description of the CNTL structure, we can easily identify the entities set and the relationships sets. The entities set consists of all system components described above including specialized entities. The specification explicitly indicates semantics information for all relationships in the three relationships sets. For example, the CONTROLLER is related to the AD_CONVERTER, the PC, and the DA_CONVERTER on its aspect relationship. Attributes, their domains, and a primary key for each relation are already described in section 5.1. The sequences of the PSQLES shown in Fig. 2 realizes the system entity structure for the CNTL. Relations representing the system entity structure are given in Fig. 3 except for the couplings relation which we did not include just because of space.

```
CREATE TABLE ER
{ EntNumber : string; EntName : string; IsComponent : Boolean;
  PRIMARY KEY : EntNumber };

INSERT INTO TABLE ER WITH VALUES { ENT1; CNTL; FALSE };
INSERT INTO TABLE ER WITH VALUES { ENT2; PLANT; TRUE };
INSERT INTO TABLE ER WITH VALUES { ENT3; SENSOR; TRUE };
INSERT INTO TABLE ER WITH VALUES { ENT4; POWER_AMP;
TRUE };
INSERT INTO TABLE ER WITH VALUES { ENT5; CONTROLLER;
TRUE };
INSERT INTO TABLE ER WITH VALUES { ENT6; MOTOR; TRUE };
.....

CREATE TABLE ASPR
{ AspName : string; EntName : string; CompEntName : string;
  PRIMAEY KEY : (AspName, CompEntName) };

INSERT INTO TABLE ASPR WITH VALUES { CNTL_ASP; CNTL;
PLANT };
INSERT INTO TABLE ASPR WITH VALUES { CNTL_ASP; CNTL;
SENSOR };
INSERT INTO TABLE ASPR WITH VALUES { CNTL_ASP; CNTL;
POWER_AMP };
.....

CREATE TABLE SPECR
{ SpecName : string; EntName : string; SpeEntName : string;
  PRIMARY KEY : (SpecName, SpecEntName) };

INSERT INTO TABLE SPECR WITH VALUES { PLANT_SPEC;
PLANT; MOTOR };
INSERT INTO TABLE SPECR WITH VALUES { PLANT_SPEC;
PLANT; AIRCON };
.....

CREATE TABLE COUPR
{ AspName : string; FromEntName : string; FromPortName : string;
ToEntName : string; ToPortName : string;
 PRIMARY KEY : (FromEntName, FromPortName) };

INSERT INTO TABLE COUPR WITH VALUES { CONTROLLER_ASP;
CONTROLLER; in; AD_CONVERTER; in};
INSERT INTO TABLE COUPR WITH VALUES { CONTROLLER_ASP;
DA_CONVERTER; analog_outout; CONTROLLER; out};
INSERT INTO TABLE COUPR WITH VALUES { CONTROLLER_ASP;
AD_CONVERTER; digital_out; PC; digital_in }
.....
```

Fig. 2. PSQLES for SES.

EntNumber	EntNaname	IsComponent
Ent1	CNTL	False
Ent2	PLANT	True
Ent3	SENSOR	True
Ent4	POWER_AMP	True
Ent5	CONTROLLER	True
Ent6	MOTOR	True
Ent7	AIRCON	True
Ent8	THERMOMETER	True
Ent9	SPEEDMETER	True
Ent10	AD_CONVERTER	True
Ent11	DA_CONVERTER	True
Ent12	PC	True

AspName	EntName	CompEntName
CNTL_ASP	CNTL	PLANT
CNTL_ASP	CNTL	SENSOR
CNTL_ASP	CNTL	POWER_AMP
CNTL_ASP	CNTL	CONTROLLER
CONTROLLER_ASP	CONTROLLER	AD_CONVERTER
CONTROLLER_ASP	CONTROLLER	DA_CONVERTER
CONTROLLER_ASP	CONTROLLER	PC

SpecName	EntName	SpecEntName
PLANT_SPEC	PLANT	MOTOR
PLANT_SPEC	PLANT	AIRCON
SENSOR_SPEC	SENSOR	THERMOMETER
SENSOR_SPEC	SENSOR	SPEEDMETER

Fig. 3. Relations for SES.
ER, ASPR, and SPECR from Top Down.

7. CONCLUDING REMARKS

The system entity structure formalism was represented by a collection of four relations and realized in a relational database. The four relations were the entities relation, the aspects relationship relation, the specializations relationship relation, and the couplings relationship relation. Representing the system entity structure as such relations made it possible to employ the well-established relational database technology as an implementation tool. Thus, operations and queries defined on a relational database can be used for manipulating a system entity structure. Moreover, efficiency in saving and retrieving a system entity structure in an external storage can be inherited from the underlying relational database. By combining an appropriate simulation environment, the realization shown in this paper would serve as a powerful model base manage-

ment system using the relational database technology. Such combination would provide a powerful environment for complex systems design based on the multifaceted modeling and simulation methodology.

ACKNOWLEDGEMENT

This investigation was supported by University of Kansas General Research allocation #3779-20-0038.

REFERENCES

Chen, P.P (1976), "The Entity-Relationship Model–Toward a Unified View of Data," *ACM Trans. on Database Systems*, vol. 1, no. 1, pp. 9-36, March.

Date, C.J. (1990), *An Introduction To Database Systems: Volume I.* Reading, MA: Addison-Wesley Publishing Company, Inc.

Higa, K. H. (1988), "End-user Logical Database Design: The Structure Entity Model Approach", Doctoral Dissertation, MIS Dept., University of Arizona.

Kim, Tag Gon and B.P. Zeigler (1990), "The DEVS-Scheme Simulation and Modelling Environment," **Chapter 2** in *Knowledge Based Simulation: Methodology and Application* (eds: Paul A. Fishwick and Richard B. Modjeski) Springer Verlag., Inc.

Kim, Tag Gon et. al. (1990), "Entity Structuring and Model Base Management," *IEEE Trans on Systems, Man and Cybernetics,* vol. 20, no. 5, pp. 1013 - 1024, September/Octorber.

Kim, Tag Gon and B.P. Zeigler (1989), "ESP-Scheme: A Realization of System Entity Structure in a LISP Environment," in *Advances in AI and Simulation*, Simulation Series, vol. 20, no. 4, pp. 135-140, March.

Rozenblit, Jerzy et. al. (1990), "Knowledge-Based System Design/Simulation Environment: Foundations and Concepts," *Journal of Operations Research Society*, vol. 41, no. 6, pp. 475-489.

Zeigler, B.P. (1984), *Multifaceted Modelling and Discrete Event Simulation.* London, UK and Orlando, FL: Academic Press.