A Procedure of Conversion of Relational into Multidimensional Database Schema

Mladen Varga

Faculty of Economics, University of Zagreb, Croatia

It is universally recognized that operational information systems lean on the relational model and data warehouses on the multidimensional model. The phrase On-Line Analytical Processing (OLAP) means summarizing, consolidating, viewing, and synthesizing data according to multiple dimensions. The process of modeling data warehouse may start from operational system's database. It may be helpful to convert a relational database schema into a multidimensional database schema in order to discover dimensions that are hidden in a relational database. However, only a few efforts have been done investigating the conversion of relational into multidimensional database schema. This paper proposes the general procedure of such a conversion. The procedure can be partly automated because some decisions of attribute type must be made during conversion.

Keywords: relational database, relational schema, multidimensional database, multidimensional schema, conversion.

1. Introduction

Operational information systems that lean on the relational model may be a starting point for the process of modeling data warehouse. Most data warehouses rely on the multidimensional model and it may be helpful to convert a relational database schema into a multidimensional database schema.

In this section, some general facts of relational and dimensional model are mentioned. In Section 2 the procedure of conversion is presented and in Section 3 a small example of the conversion is shown.

1.1. Relational Model

A relational database consists of a set of relations. A relational schema which is used to describe a relation r, denoted by $R(A_1, A_2, \ldots, A_n)$ is made up of a relation named R and a list of attributes $A_1, A_2, \ldots, A_n$. Each attribute A_i is the name of a role played by some domain D in the relation R. A relation r of the relational schema $R(A_1, A_2, \ldots, A_n)$ is a set of tuples $r = \{t_1, t_2, \ldots, t_m\}$. A relational database is described by a relational database schema that consists of a set of relational schemas.

If for any two distinct tuples t_1 and t_2 in the relation r of R there exists attribute (set of attributes) K such that $t_1[K] \neq t_2[K]$, then such attribute (set of attributes) is called key. It is common to designate one of possible keys as the primary key, which is used to identify tuples in the relation.

A set of attributes FK in the relational schema R is a foreign key of R_1 if two rules are satisfied:

- The attributes in FK have the same domain as the primary key PK of another relational schema attributes FK are said to reference relation R_2,
- A value of FK in a tuple t_1 of R_1 either occurs as a value of PK for some tuple t_2 in R_2 or is null.

1.2. Dimensional Model

Let us suppose that a hospital database contains admission data (such as number of days and

value) of the patient, the date of admission, and the diagnosis. The admission data is determined by three attributes Patient, Diagnosis and Time. They are referred to as *dimensions* while the determined admission attributes are referred to as *measures* or *facts* or *fact attributes*. The facts are mostly numerical, preferably continuously valued and additive. They vary over time. In the statistical database field (Shosani, 1997) the dimension corresponds to *category attribute*, and the measure to *summary attribute*. The cube model, shown in Fig. 1, is very useful in presenting the concept of the multidimensional space. There is no a priori distinction between dimensions and measures while any attribute can play either role (Gyssens, 1996). There is no formal way to decide which attributes are dimensions and which attributes are measures. This decision has to be solved during database design.

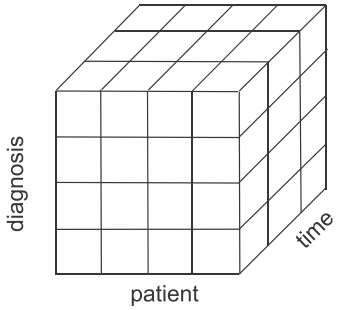


Fig. 1. Cube model.

The structure of *the dimensional model* can be represented by the *star join schema* (Kimball, 1996) shown in Fig. 2. The center of the schema is the *fact table*, which is the only table in the schema with multiple joins connecting it to other tables. The fact table is where the facts of the business are stored (Kimball, 1996) such as NumberOfDays and Value in the hospital database. The other tables are the *dimension tables*. Dimension attributes describe the item in the dimension, and are virtually constant over time. The primary key of the fact table is composite or concatenated key, which is the combination of as many foreign keys as many dimensions there are in the schema. Each component of the composite key is a foreign key referencing the primary key of a dimension table. In other words, every fact table represents a many-to-many relationship, that is, it contains as many foreign keys as many dimensions there are in the schema. A multidimensional database may consist of any number of star join schemas with some dimension tables overlapping.

According to (Lehner, 1998) we shall introduce some definitions which will help placing and solving the problem of conversion of relational database schema into multidimensional database schema.

The notion of functional dependency is of fundamental importance: If A and B are (sets of) attributes in a database, B is functionally dependent on A or A functionally determines B, denoted by $A \rightarrow B$, if and only if for each specific value $a \in A$, there exists at most one value $b \in B$.

Dimensions are usually organized into *hierarchies* that specify aggregation level and hence granularity of viewing data. A dimension is defined over a dimensional schema. A dimensional schema is a set of dimensional attributes $\mathsf{D} = \{D_1, \ldots, D_k\}$ where for each $D_i \in \mathsf{D}$ there exists a $D_j \in \mathsf{D} - \{D_i\}$ such that either $D_i \rightarrow D_j$ or $D_j \rightarrow D_i$. A categorization C of a dimensional schema D is a sequence of functional de-

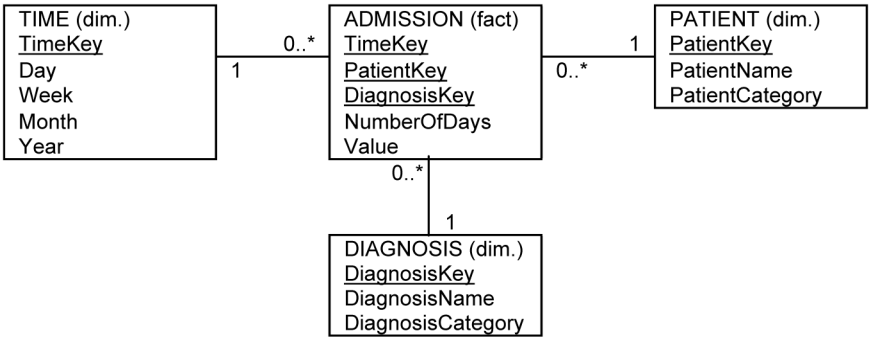


Fig. 2. Star join schema.

pendencies $D_1 \rightarrow D_2, \ldots, D_{k-1} \rightarrow D_k$ where $D_i \in \mathsf{D}$ for each $i = 1, \ldots, k$. The attribute D_i is the i^{th} categorization level. A dimensional attribute $D_t \in \mathsf{D}$ is called *terminal*, if there is no $D \in \mathsf{D} - \{D_t\}$ such that $D \rightarrow D_t$.

Example:

A simple dimensional schema of the product dimension consists of the dimension attributes $\mathsf{Product} = \{\mathsf{Product}, \mathsf{ProductGroup}, \mathsf{ProductType}\}$. The hierarchy on the product dimension $\mathsf{Product} \rightarrow \mathsf{ProductGroup} \rightarrow \mathsf{ProductType}$ is a set of functionally interrelated dimensional attributes Product, ProductGroup and ProductType. The attribute Product is terminal and belongs to 1st category level, ProductGroup to 2nd and ProductType to 3rd category level.

A multidimensional database is defined over a set of multidimensional schemas. A multidimensional schema $\mathsf{M} = (\{\mathsf{D}_1, \ldots, \mathsf{D}_n\}, S)$ consists of a set of dimensional schemas $\{\mathsf{D}_1, \ldots, \mathsf{D}_n\}$, measures or facts S and a functional dependency $\{\mathsf{D}_1, \ldots, \mathsf{D}_n\} \rightarrow S$. Dimension tables are described by dimensional schemas and a fact table by the dependency $\{\mathsf{D}_1, \ldots, \mathsf{D}_n\} \rightarrow S$.

A dimension defined by a dimensional schema D is in *dimensional normal form* (DNF) if the following conditions are satisfied:

- There exists exactly one terminal dimension attribute $D_t \in \mathsf{D}$
- The values of D_t are complete, i.e. each object has to be assigned to some value of D_t (see explanation of the completeness condition in Section on Summarizability)

A multidimensional schema $\mathsf{M} = (\{\mathsf{D}_1, \ldots, \mathsf{D}_n\}, S)$ is in multidimensional normal form if the following conditions are satisfied:

- Each D_i is in dimensional normal form
- The dimensions are orthogonal to each other, i.e. there exists no functional dependency $C \rightarrow D$ where $C \in \mathsf{D}_i$ and $D \in \mathsf{D}_j$ and $i \neq j$
- The fact attribute S is fully functionally determined by the set of the terminal dimension attributes of the dimensions

Summarizability of Fact Attributes

Multidimensional databases are mostly used to perform statistical analysis. Summarizability of fact attributes is an extremely important property of a multidimensional schema. In general, summarizability (Lehner, 1998) is needed to ensure correct automatic aggregation operations that are important in OLAP drill-down and roll-up operations. An incorrect summarization that is not obviously seen can lead to erroneous conclusions. The following example (Lenz, Shoshani, 1997) illustrates the problem.

	Year			
Dept.	1997	1998	1999	Totals
Math	14	20	18	32
Chemistry	10	12	9	19
Totals	24	32	27	51

Fig. 3. Summarizability example.

Data in Fig. 3 provide a count of the number of university students organized by department and year. Suppose we wish to find out the number of students that attended each department over the whole period of 3 years. If students attended the department for more than one year, then adding the counts for each year is incorrect. The total for math department $14+20+18 = 52$ is incorrect. If the math department started in 1997 there were 14 students in 1997 who also attended it in 1998. Consequently, there were only 6 new students in 1998. These 6 students continued in 1999, and 12 new students joined for a total of 18. Thus, the total number of students who attended the math dept. over 3-year period is $14 + 6 + 12 = 32$. Likewise, for chemistry dept. it is: $10 + 2 + 7 = 19$. The problem is more complicated and, in general, it is not possible to compute summaries for a non-summarizable fact because the semantic rule may not be known. In the example above, it is assumed that all students attend a 2-year program but in reality this is not true. Some students attend more than 2 years some drop after the first year. Thus, for non-summarizable fact the totals can be extracted only from the base data, which includes the information about each individual student. This non-summarizability of the number of students is not absolute for both dimensions. The number of students is summarizable to the department dimension. In the year 1997 there were 24 students, 14 in the math department and 10 in the chemistry department.

The summarizability problem is more complicated than shown in the previous example. (Lenz, Shoshani, 1997) discuss problem more thoroughly and argue that three conditions are sufficient for solving summarizability problem. The necessary conditions for summarizability are:

- Disjointness of dimensions (dimension attributes)
- Completeness
- Type compatibility of function and fact attribute

Disjointness of dimension attributes states that the dimension attributes must form disjoint subsets. This condition is satisfied if categorization of a dimensional schema D is a sequence of functional dependencies $D_1 \rightarrow D_2, \ldots, D_{k-1} \rightarrow D_k$ where $D_i \in D$ for each $i = 1, \ldots, k$. The example is the hierarchy on the Time dimension Day→ Month→Year. The year 1999 consists of a disjoint set of month attribute: Jan 1999, Feb 1999,. . . , Dec 1999. The month Jan 1999 consists of a disjoint set of day attribute: 1st day of Jan 1999, 2nd day of Jan 1999 etc.

Once the disjointness condition is satisfied, it is necessary to test whether the grouping of objects into clusters is complete. First, all elements of the cluster must exist, i.e. the union of all clusters must constitute the entire set of objects. There must be no missing object, i.e. the object that does not belong to a cluster. In other words, *completeness* states that each object must be assigned to some category. For example, if some product is not assigned to a product group, the condition is violated. Second, all members of subset values must exist and their union must constitute the entire set. Fig. 4 shows examples of violation of the disjointness and the completeness conditions.

Type compatibility condition depends on the type of fact attribute as well as on the statistical function applied. For every fact attribute and for every dimension it must be determined whether summarization is allowed or not. Fact attributes (Kimball, 1996; Lenz, Shoshani, 1997) may be classified as:

- *additive*, if it is additive along all dimensions (such as *flow* or *rate*): annual income, weekly number of rainy days
- *semiadditive*, if it is not additive along one or more dimensions, usually temporal dimension (such as *level* or *stock*): number of students in a class, inventory of goods
- *nonadditive*, if it is additive along no dimension (such as *value-per-unit*): item price

Flow refers to periods and is recorded at the end of each period. It records some cumulative effect over a period. *Level* is measured and recorded at a particular point of time. It records the state at the specific point in time. *Value-per-unit* is usually the current value of the measured unit.

The most important function is sum. Sum is prohibited to value-per-unit data in general and likewise to level data along temporal dimensions. Therefore, summarization is performed differently to temporal dimensions as opposed to non-temporal dimensions. Next table (Fig. 5) (Lenz, Shoshani, 1997) shows how the type of attribute affects the sum function along temporal and non-temporal dimensions.

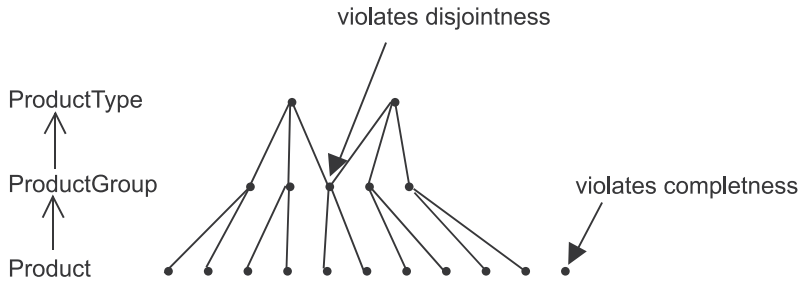


Fig. 4. Examples of violation of the disjointness and the completeness conditions.

		Fact attribute		
		additive (flow, rate)	semiadditive (level, stock)	nonadditive (value-per-unit)
Dimension	temporal	✓	×	×
	non-temporal	✓	✓	×

Fig. 5. Sum function along temporal and non-temporal dimensions.

In the design phase of a multidimensional schema the type of the dimension (temporal, non-temporal), the type of fact attribute (additive: flow or rate, semiadditive: level or stock, nonadditive: value-per-unit) has to be determined. (Lehner et al., 1998) suggest that in the graphical notation every temporal dimension is marked by a dot.

If we look at the definition of the multidimensional normal form, we can see that it ensures disjointness of dimension attributes and completeness property. However, it does not automatically ensure type compatibility because it is dynamic property. Therefore, type compatibility cannot be checked in the static conditions of multidimensional normal form but in the dynamic conditions when statistical operations are performed.

2. Procedure of Conversion

The conversion of a relational into a multidimensional database schema has the following goals:

- *Find all dimensions* that exist but are hidden in a relational database schema.
- Get the multidimensional database schema that is *equally expressive* as its initial relational schema, i.e. the multidimensional schema must reflect the same application knowledge.
- The resulting multidimensional schema must be in the multidimensional normal form *to ensure summarizability* (Lehner et al., 1998).

Input relational database schema *must be normalized*. Since this procedure is based on the notion of functional dependencies the necessary normal form is Boyce-Codd normal form (BCNF). A relation R is in BCNF if whenever a functional dependency $X \rightarrow A$ holds in R, X is a superkey of R.

For the sake of clarity in the following procedure we shall use the term *relation* in the relational model and the term *table* for the same entity in the dimensional model. We assume that all primary and foreign keys of relational schema are known.

The procedure of conversion of a relational into a multidimensional database schema consists of these steps:

A. Discovering potential dimensions

B. Defining fact and dimension attributes

C. Defining fact and dimension tables

D. Defining attributes in fact tables

E. Defining attributes in dimension tables

F. Checking multidimensional schema(s)

A. Discovering Potential Dimensions

Each row in a relation represents an *object instance*, which is an instance of a simple object (such as PATIENT object) or an instance of a composition of simple objects (such as RESERVATION which is the relationship between PERSON, HOTEL and DATE object instances). The object instance's properties are represented by object's *attributes*. Primary key uniquely identifies the object instance while the other attributes refer to mapping instances that the object instance has with instances of the other objects (Martin, Odell, 1995). Foreign keys refer to mapping instances that the object instance has with instances of existing objects. Nonkey attributes refer to mapping instances that the object instance has with nonexisting objects instances and play the role of property attributes of the object. For example, the Date attribute in the OPERATION relation represents the date of operation performed on the patient. It represents the mapping of the OPERATION instance object to

the DATE object instance. Usually, in the operational (transactional) databases the DATE object does not exist so the DATE relation does not exist either. Therefore, the Date attribute becomes the attribute of the OPERATION object. In data warehouses the Date is an existing object, called dimension, having the property attributes such as Week, WeekendFlag, HolidayFlag, Season, Month and Year in the dimension relation.

Looking for nonkey attributes in a relational schema, i.e. attributes that are not part of primary or foreign keys, we may discover potential foreign keys to hidden dimensions, i.e. dimensions that have no dimension relation. We may open such a hidden dimension by creating a new dimension relation. The potential foreign key becomes the real foreign key referencing the primary key of newly created dimension relation.

Example:

Let us assume a relation described by its relational schema

ADMISSION(Admission#, *Ward#*(WARD.Ward#), *Doctor#*(DOCTOR.Doctor#), Date, NumberOfDays, Value, Result)

where the primary key is underlined and foreign keys are italicized and followed by the referenced relation and its primary key (referential integrity of foreign key to its primary key).

In the ADMISSION relation we have discovered a potential foreign key Date to the time dimension. We may open this dimension by creating the new TIME relation with properties (attributes) Date, Month, Quarter and Year. The attribute Date in the ADMISSION relation becomes a foreign key to the new TIME relation. The result is

ADMISSION(Admission#, *Ward#*(WARD.Ward#), *Doctor#*(DOCTOR.Doctor#), *Date*(TIME.Date), NumberOfDays, Value, Result)
TIME(Date, Month, Quarter, Year)

At the same time we may define hierarchy on the time dimension by functional dependencies *Date*→*Month*→*Quarter*→*Year*.

B. Defining Fact and Dimension Attributes

By inspecting all nonkey attributes in every relational schema, i.e. the attributes that are neither primary nor foreign keys, we shall define the nonkey attributes that are fact attributes and the nonkey attributes that are dimension attributes. This is the key decision while there is no formal way to decide which attributes are fact attributes. If an attribute value shows some fact of the business, varies over time, and is preferably continuosly valued, it is a strong candidate for a fact attribute.

Example:

Let us assume relations

ORDER-ITEM(Order#, Product#, Quantity)
PRODUCT(Product#, ProductName)

There are two nonkey attributes: the Quantity in the ORDER-ITEM relation and the ProductName in the PRODUCT relation. At this step we must decide whether these attributes are fact attributes or dimension attributes. The Quantity attribute shows a fact or measure of the business, the number of products ordered. The attribute value is continuosly valued and summarizable, therefore the Quantity is a fact attribute. On the other hand, the ProductName attribute is not a measure. It is a property attribute of the product, and the dimension attribute of the product dimension.

Nonkey attributes in a normalized relation are all fact or all dimension attributes. A normalized relation represents either an object where all nonkey attributes are property attributes of the object or a composition of two or more objects where nonkey attributes are property attributes of the relationship of these objects. The former attributes are dimension attributes, the latter are fact attributes.

C. Defining Fact and Dimension Tables

Each relation must be considered individually:

- If there exists any fact attribute in the relation, the fact table will be created.
- If there exists no fact attribute in the relation and the relation has more than one foreign key corresponding to many-to-many relationship, a fact table will be created. This is a factless fact table, i.e. a fact table that has no fact attribute.
- If there exists any dimension attribute in the relation, the dimension table will be created.

D. Defining Attributes in Fact Tables

For each relation that gives a fact table all foreign keys, primary keys and nonkey attributes must be particularly considered in the following procedure.

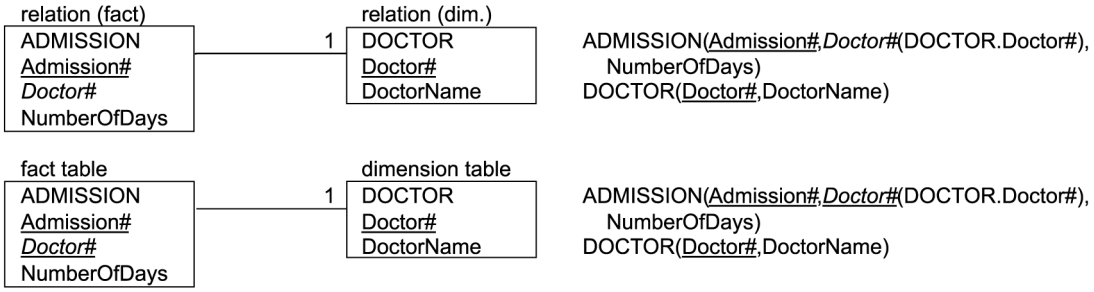
1. Foreign keys of the relation
Either step 1.1 or step 1.2 is executed:

1.1 If a foreign key of the relation references the primary key of the relation that gives a dimension table, the following options are possible:

a) The foreign key becomes the primary key of the fact table. At the same time it references the primary key of the corresponding dimension table.

This rule defines *existing dimension with existing dimension table*.

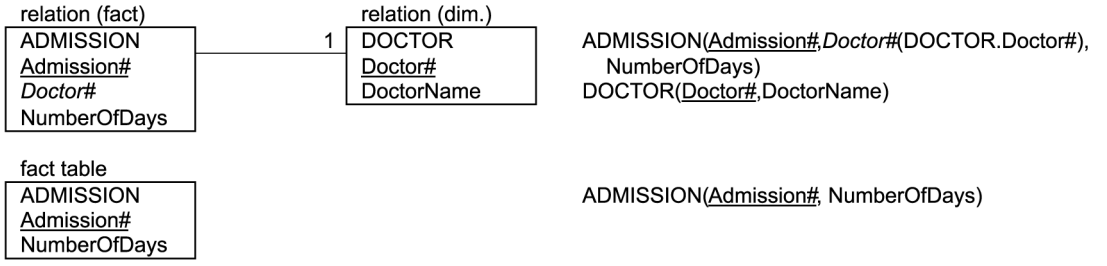
In the following example relations of the relational schema are in the first row, and tables in the dimensional schema are in the second row. We are considering primarily the fact table in the first column of the figure and the transformation of its foreign keys. Relationships between the relations or tables are shown in UML cardinality notation. Here, one admission is required by one and only one doctor. The foreign key of the ADMISSION relation is Doctor#. It becomes part of the primary key in the ADMISSION fact table and remains the foreign key referencing the dimension table DOCTOR.



b) The foreign key is dropped from the fact table. In this way the dimension is dropped. The corresponding dimension table must be dropped if at the end of the conversion process there are no foreign keys referencing this dimension table.

This rule defines *dropping existing dimension and its corresponding dimension table*.

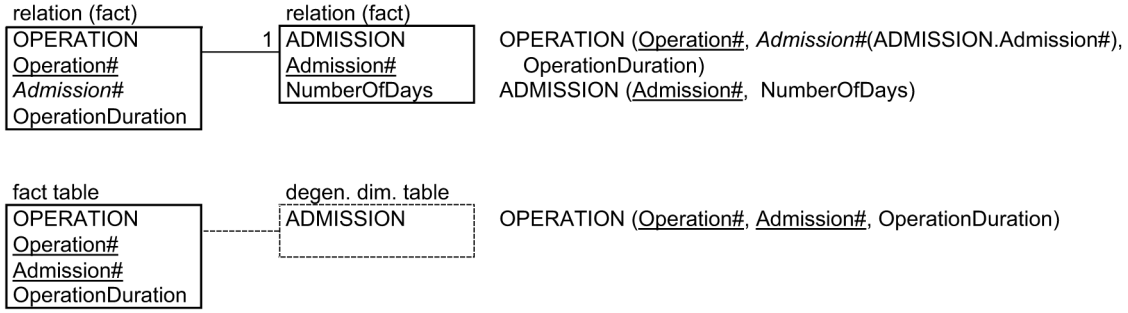
Here is the previous example with the existing dimension dropped.



1.2 If the foreign key of the relation references the primary key of the relation that gives a fact table, the foreign key becomes the primary key of the fact table. But its foreign key role must be canceled because the primary key it references to is changed after step **D** of the procedure is applied to the primary key's fact table.

This rule defines *finding and defining a degenerate dimension*.

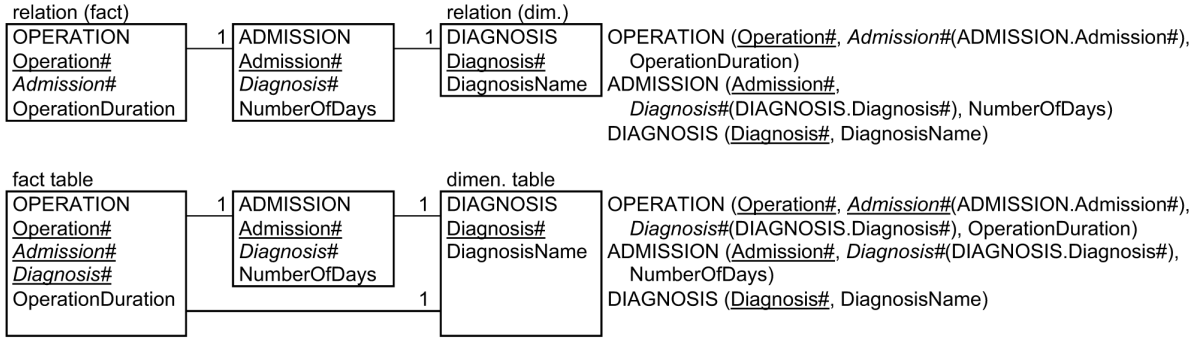
In the following example the Admission# foreign key in the OPERATION relation are changed to primary key of the OPERATION fact table. Its foreign key role is canceled.



1.3 After step 1.1 or 1.2 is executed all foreign keys of the referenced relation must be removed to the considered fact table and treated as foreign keys described in this step (step 1.).

This rule defines *finding and defining new dimensions*.

In the following example after the Admission# foreign key in the OPERATION relation has become the primary key in the OPERATION fact table all foreign keys of the ADMISSION relation are removed to the OPERATION fact table. Here the only foreign key removed is the Diagnosis#. It references the primary key of the DIAGNOSIS dimension table, so the new DIAGNOSIS dimension table is defined and connected to the OPERATION fact table.



The explanation of foreign keys transition to the considered fact table is based on applying the transition rule among functional dependencies on considered keys before the foreign key transition is applied. In the example mentioned, functional dependencies are as follows: OPERATION.Operation# → OPERATION.Admission# (primary key functionally determines nonkey attribute in the same relation because the relation is in 3NF), OPERATION.Admission# → ADMISSION.Admission# (referential integrity of the Admission# foreign key in the OPERATION relation to the Admission# primary key in the ADMISSION relation), ADMISSION.Admission → ADMISSION.Diagnosis# (primary key functionally determines nonkey attribute in the same relation), and ADMISSION.Diagnosis# → DIAGNOSIS.Diagnosis# (referential integrity of the Diagnosis# foreign key in the ADMISSION relation to the Diagnosis# primary key in the DIAGNOSIS relation). Consequently, by applying transition rules: OPERATION.Operation# → DIAGNO-

SIS.Diagnosis#. The same functional dependency exists after the Diagnosis# foreign key is removed from the ADMISSION relation into the OPERATION fact table. Now the diagnosis dimension is a dimension which is achieved from OPERATION fact table directly and not transitively as was the case with foreign key.

2. The primary key, i.e. every attribute that is part of the primary key and is not part of the foreign key of the relation becomes the primary key of the fact table without referencing to any dimension table.

This rule defines *finding and defining a degenerate dimension* (Kimball, 1996).

Identifiers of many documents are of such a nature, such as Order# and Ticket#. It is important not to drop these attributes in order to retain the expressiveness of dimensional schema equal to the relational schema.

3. Nonkey attributes may be:

3.1 dropped from further processing, i.e. will not be included into the fact table.

3.2 included into the fact table as fact attributes.

E. Defining Attributes in Dimension Tables

All foreign keys may be dropped from the dimension tables because they are not necessary after step **D**.1.3 is applied. In this step all possible foreign keys from referenced relations are removed to the considered fact table. Therefore, all necessary foreign keys are after step **D**.1.3 settled in each fact table and there is no need to use foreign keys of dimension tables.

The attributes that belong to the primary key or are nonkey attributes retain their roles.

F. Checking Multidimensional Schema(s)

Every multidimensional schema must be considered for its expressiveness, normal form and summarizability.

Expressiveness of Multidimensional Schema

The expressiveness of multidimensional schema is retained if steps **D**.1.1b and **D**.3.1 are not applied while in these steps some information is dropped.

Multidimensional Normal Form

The procedure does not automatically guarantee that multidimensional database schema is in the multidimensional normal form. We must manually check every dimensional schema to ensure that it satisfies dimensional normal form, i.e. that there exists exactly one terminal attribute, and that its values are complete.

Summarizability

The type of every fact attribute (flow, level or value-per-unit) must be specified and declared in the multidimensional schema. Also, in the graphical notation all temporal dimensions must be specified and marked by a dot.

3. Example

The entity-relationship model of a simplified hospital example, similar to the example in Golfarelli (1998a), is shown in Fig. 6. Admission (entity ADMISSION) of a patient (entity PATIENT and TOWN) to a hospital is described by diagnosis (DIAGNOSIS, ADMISSION-SECONDARY-DIAGNOSIS), a ward in the hospital (WARD) where a patient is transfered, operation (OPERATION, OP-THEATRE) that was performed on a patient and doctors (DOCTOR) who diagnose, cure and operate on a patient. Primary keys are underlined and foreign keys are italicized. In such a diagram foreign keys are not entity attributes. They are the relational implementation of entity relationships and they are here only for the sake of clarity of the problem.

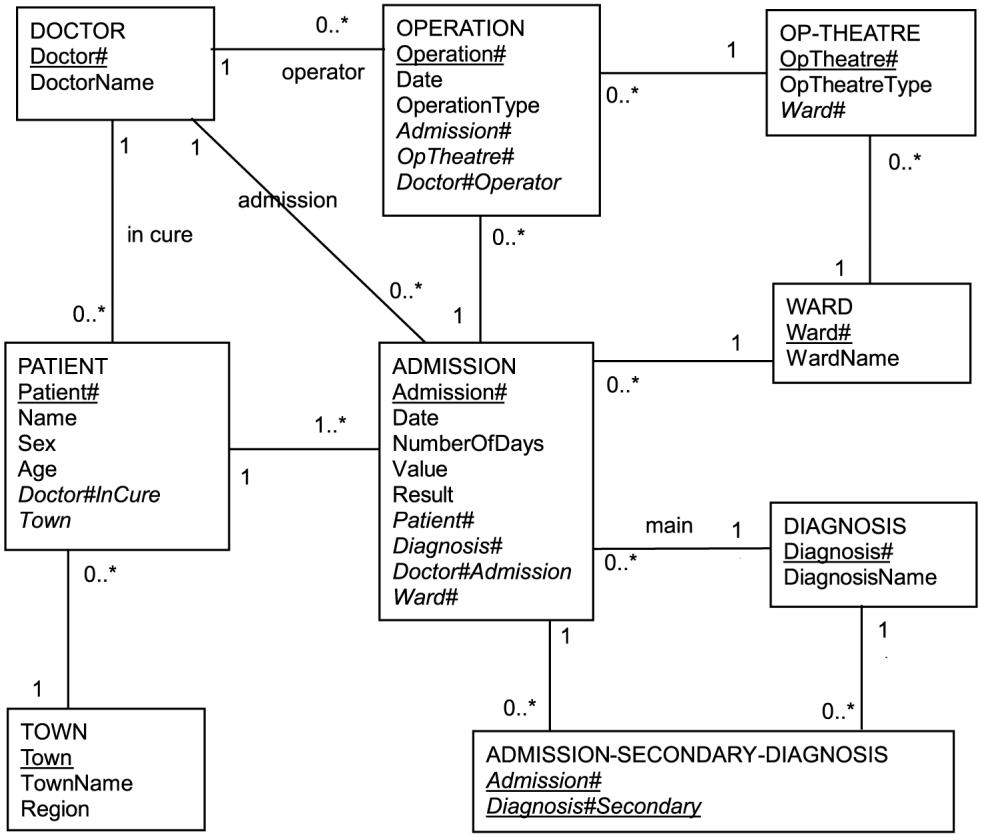


Fig. 6. Hospital example.

Its relational database schema is:

DOCTOR(Doctor#, DoctorName)
OPERATION(Operation#, Date, OperationType,
Admission#(ADMISSION.Admission#),
OpTheatre#(OP-THEATRE.OpTheatre#),
Doctor#Operator(DOCTOR.Doctor#))
PATIENT(Patient#, Name, Sex, Age,
Doctor#InCure(DOCTOR.Doctor#),
Town(TOWN.Town))
ADMISSION(Admission#, Date, NumberOfDays,
Value, Result, *Patient#*(PATIENT.Patient#),
Diagnosis#(DIAGNOSIS.Diagnosis#),
Doctor#Admission(DOCTOR.Doctor#),
Ward#(WARD.Ward#))
WARD(Ward#, WardName)
OP-THEATRE(OpTheatre#, OpTheatreType,
Ward#(WARD.Ward#))
DIAGNOSIS(Diagnosis#, DiagnosisName)
TOWN(Town, TownName, Region)
ADMISSION-SECONDARY-DIAGNOSIS
(*Admission#*(ADMISSION.Admission#),
Diagnosis#Secondary(DIAGNOSIS.Diagnosis#))

The procedure of conversion of its relational database into a multidimensional database schema follows.

A. Discovering Potential Dimensions

In some relations we may find few foreign keys to potential dimensions:

Relation	Foreign key to potential dimension	Dimension
OPERATION	Date	TIME
PATIENT	Sex, Age	SEX, AGE
ADMISSION	Date, Result	TIME, RESULT

The new relations are created:

TIME(Date, Month, Quarter, Year) with Date→Month →Quarter→Year hierarchy
SEX(Sex, Name)
AGE(Age, Age10)
RESULT(Result, ResultDescription)

Now, the relational schema consists of the following relations:

DOCTOR(Doctor#, DoctorName)
OPERATION(Operation#, *Date*(TIME.Date),
OperationType,
Admission#(ADMISSION.Admission#),
OpTheatre#(OP-THEATRE.OpTheatre#),
Doctor#Operator(DOCTOR.Doctor#))
PATIENT(Patient#, Name, *Sex*(SEX.Sex),
Age(AGE.Age),
Doctor#InCure(DOCTOR.Doctor#),
Town(TOWN.Town))
ADMISSION(Admission#, *Date*(TIME.Date),
NumberOfDays, Value,
Result(RESULT.Result),
Patient#(PATIENT.Patient#),
Diagnosis#(DIAGNOSIS.Diagnosis#),
Doctor#Admission(DOCTOR.Doctor#),
Ward#(WARD.Ward#))
WARD(Ward#, WardName)
OP-THEATRE(OpTheatre#, OpTheatreType,
Ward#(WARD.Ward#))
DIAGNOSIS(Diagnosis#, DiagnosisName)
TOWN(Town, TownName, Region)
ADMISSION-SECONDARY-DIAGNOSIS
(*Admission#*(ADMISSION.Admission#),
Diagnosis#Secondary(DIAGNOSIS.Diagnosis#))
TIME(Date, Month, Quarter, Year)
SEX(Sex, Name)
AGE(Age, Age10)
RESULT(Result, ResultDescription)

B. Defining Fact and Dimension Attributes

There is no formal way to decide which attributes are fact attributes and which attributes are dimension attributes because nonkey attributes in the relational schema can play either role. Inspecting all nonkey attributes in the relational schema we should manually decide which attributes are fact attributes and which attributes are dimension attributes. The fact attributes show measures of the business. They are mostly numerical, continuously valued and additive. In the example they are depicted in light gray. The summarizability type (flow, level or value-per-unit) of the fact attribute is also determined. Dimension attributes are property attributes of dimensions. In the example they are depicted in dark gray.

DOCTOR(Doctor#, DoctorName)
OPERATION(Operation#, *Date*(TIME.Date),
OperationType:value-per-unit,
Admission#(ADMISSION.Admission#),
OpTheatre#(OP-THEATRE.OpTheatre#),
Doctor#Operator(DOCTOR.Doctor#))
PATIENT(Patient#, Name, *Sex*(SEX.Sex),
Age(AGE.Age),
Doctor#InCure(DOCTOR.Doctor#),
Town(TOWN.Town))
ADMISSION(Admission#, *Date*(TIME.Date),
NumberOfDays:level, Value:level,
Result(RESULT.Result),
Patient#(PATIENT.Patient#),
Diagnosis#(DIAGNOSIS.Diagnosis#),
Doctor#Admission(DOCTOR.Doctor#),
Ward#(WARD.Ward#))
WARD(Ward#, WardName)
OP-THEATRE(OpTheatre#, OpTheatreType,
Ward#(WARD.Ward#))
DIAGNOSIS(Diagnosis#, DiagnosisName)
TOWN(Town, TownName, Region)
ADMISSION-SECONDARY-DIAGNOSIS
(*Admission#*(ADMISSION.Admission#),
Diagnosis#Secondary(DIAGNOSIS.Diagnosis#))

Obviously the attributes of new dimensions are dimension attributes:

TIME(Date, Week, Quarter, Year)
SEX(Sex, Name)
AGE(Age, Age10)
RESULT(Result, ResultDescription)

C. Defining Fact and Dimension Tables

Because of the fact attributes the OPERATION and ADMISSION fact tables are created. In the ADMISSION-SECONDARY-DIAGNOSIS relation there is no fact attribute, but there are two foreign keys that correspond to many-to-many relationship. Therefore, the fact table with no fact attribute is created. Relations DOCTOR, PATIENT, WARD, OP-THEATRE, DIAGNOSIS, TOWN, TIME, SEX, AGE and RESULT give dimension tables.

D. Defining Attributes in Fact Tables

In the relation

OPERATION(Operation#, *Date*(TIME.Date),
OperationType:value-per-unit,

Admission#(ADMISSION.Admission#),
OpTheatre#(OP-THEATRE.OpTheatre#),
Doctor#Operator(DOCTOR.Doctor#))

the attributes are as follows, with the following rules applied:

— By rule 2 the primary key Operation# becomes the primary key of the degenerate operation dimension.

— By rule 1.1a foreign key *Date* becomes the primary key of the fact table and foreign key referencing TIME.Date. It is renamed in *OperationDate*.

— By rule 3.2 OperationType is the fact attribute of value-per-unit type.

— By rule 1.2 foreign key *Admission#* becomes the primary key of the fact table and foreign key to degenerate admission dimension.

— By rule 1.1a foreign key *OpTheatre#* becomes the primary key of the fact table and foreign key referencing OP-THEATRE.OpTheatre#.

— By rule 1.1a foreign key *Doctor#Operator* becomes the primary key of the fact table and foreign key referencing DOCTOR.Doctor#.

— By rule 1.3 all foreign keys of the referenced relations are removed to the OPERATION fact table. From the ADMISSION relation *Date*, *Result*, *Doctor#Admission*, *Patient#*, *Ward#* and *Diagnosis#* are removed. By rule 1.1a *Date* becomes the primary key of the fact table and foreign key referencing TIME.Date. It is renamed to *AdmissionDate*. By the same rule *Result* becomes the primary key and foreign key referencing RESULT.Result, *Doctor#Admission* (doctor who admits the patient to hospital) becomes the primary key and foreign key referencing DOCTOR.Doctor#, *Patient#* becomes the primary key and foreign key referencing PATIENT.Patient#, *Ward#* (the hospital ward where the patient is transferred) becomes the primary key and foreign key referencing WARD.Ward# and *Diagnosis#* (main diagnosis) becomes the primary key and foreign key referencing DIAGNOSIS.Diagnosis#. *Ward#* (the ward where the operation on the patient is undertaken) is removed from the OP-THEATRE. *Ward#* is renamed to *OperationWard#* which becomes primary key and foreign key to WARD.Ward#. From the PATIENT relation removed are *Sex*, *Age*, *Doctor#InCure* and *Town*. By rule 1.1a *Sex* becomes the primary key and foreign key referencing SEX.Sex. By the same rule *Age* becomes primary key and foreign key referencing AGE.Age, *Doctor#InCure* becomes primary key and foreign key referencing DOCTOR.Doctor# and *Town* primary key and foreign key referencing to TOWN.Town.

The fact table OPERATION is now:

OPERATION(Operation#, *OperationDate*(TIME.Date),
OperationType:value-per-unit, Admission#,
OpTheatre#(OP-THEATRE.OpTheatre#),
Doctor#Operator(DOCTOR.Doctor#),
AdmissionDate(TIME.Date),
Result(RESULT.Result),
Doctor#Admission(DOCTOR.Doctor#),
Patient#(PATIENT.Patient),
Ward#(WARD.Ward#),
Diagnosis#(DIAGNOSIS.Diagnosis#),
OperationWard#(WARD.Ward#),
Sex(SEX.Sex), *Age*(AGE.Age),
Doctor#InCure(DOCTOR.Doctor#),
Town(TOWN.Town),)

In the relation

ADMISSION(Admission#, *Date*(TIME.Date),
NumberOfDays:level, Value:level,
Result(RESULT.Result),
Patient#(PATIENT.Patient#),
Diagnosis#(DIAGNOSIS.Diagnosis#),
Doctor#Admission(DOCTOR.Doctor#),
Ward#(WARD.Ward#))

the attributes are as follows, with the following rules applied:

— By rule 2 the primary key Admission# becomes the primary key of the degenerate admission dimension.

— By rule 1.1a foreign key *Date* becomes the primary key of the fact table and foreign key referencing TIME.Date. It is renamed in *AdmissionDate*.

— By rule 3.2 NumberOfDays and Value are fact attributes, both of type level.

— By rule 1.1a foreign key *Result* becomes the primary key of the fact table and foreign key referencing RESULT.Result.

— By rule 1.1a foreign key *Patient#* becomes the primary key of the fact table and foreign key referencing PATIENT.Patient#.

— By rule 1.1a foreign key *Diagnosis#* becomes the primary key of the fact table and foreign key referencing DIAGNOSIS.Diagnosis#.

— By rule 1.1a foreign key *Doctor#Admission* becomes the primary key of the fact table and foreign key referencing DOCTOR.Doctor#.

— By rule 1.1a foreign key *Ward#* becomes the primary key of the fact table and foreign key referencing WARD.Ward#.

— By rule 1.3 all foreign keys of the referenced relations are removed to the ADMISSION fact table. From the PATIENT relation *Sex*, *Age*, *Doctor#InCure* and *Town* are removed. By rule 1.1a *Sex* becomes the primary key and foreign key referencing SEX.Sex. By the same rule *Age* becomes primary key and foreign key referencing AGE.Age, *Doctor#InCure* becomes primary key and foreign key referencing DOCTOR.Doctor# and *Town* primary key and foreign key referencing TOWN.Town.

Now the ADMISSION table is

ADMISSION(Admission#, *AdmissionDate*(TIME.Date),
NumberOfDays:level, Value:level,
Result(RESULT.Result),
Patient#(PATIENT.Patient#),
Diagnosis#(DIAGNOSIS.Diagnosis#),
Doctor#Admission(DOCTOR.Doctor#),
Ward#(WARD.Ward#),
Sex(SEX.Sex), *Age*(AGE.Age),
Doctor#InCure(DOCTOR.Doctor#),
Town(TOWN.Town))

In the relation

ADMISSION-SECONDARY-DIAGNOSIS
(*Admission#*(ADMISSION.Admission#),
Diagnosis#Secondary(DIAGNOSIS.Diagnosis#))

the attributes are as follows, with the following rules applied:

— By rule 1.2 foreign key *Admission#* becomes the primary key of the fact table and foreign key to degenerate admission dimension.

— By rule 1.1a foreign key *Diagnosis#Secondary* becomes the primary key of the fact table and foreign key referencing DIAGNOSIS.Diagnosis#.

— By rule 1.3 all foreign keys of the referenced relations are removed to the ADMISSION-SECONDARY-DIAGNOSIS fact table. From the ADMISSION relation *Date*, *Result*, *Doctor#Admission*, *Patient#*, *Ward#* and *Diagnosis#* are removed. By rule 1.1a *Date* becomes the primary key of the fact table and foreign key referencing TIME.Date. It is renamed in *AdmissionDate*. By the same rule *Result* becomes the primary key and foreign key referencing RESULT.Result, *Doctor#Admission* (doctor who admits the patient to hospital) becomes the primary key and foreign key referencing DOCTOR.Doctor#, *Patient#* becomes the primary key and foreign key referencing PATIENT.Patient#, *Ward#* (the hospital ward where the patient is transferred) becomes the primary key and foreign key referencing WARD.Ward# and *Diagnosis#* (main diagnosis) becomes the primary key and foreign key referencing DIAGNOSIS.Diagnosis#. From the PATIENT relation *Sex*, *Age*, *Doctor#InCure* and *Town* are removed. By rule 1.1a *Sex* becomes the primary key and foreign key referencing SEX.Sex. By the same rule *Age* becomes primary key and foreign key referencing AGE.Age, *Doctor#InCure* becomes primary key and foreign key referencing DOCTOR.Doctor# and *Town* primary key and foreign key referencing TOWN.Town.

Now the ADMISSION-SECONDARY-DIAGNOSIS table is

ADMISSION-SECONDARY-DIAGNOSIS(Admission#,
Diagnosis#Secondary(DIAGNOSIS.Diagnosis#),
AdmissionDate(TIME.Date),
Result(RESULT.Result),
Doctor#Admission(DOCTOR.Doctor#),
Patient#(PATIENT.Patient#),
Ward#(WARD.Ward#),
Diagnosis#(DIAGNOSIS.Diagnosis#),
Sex(SEX.Sex), *Age*(AGE.Age),
Doctor#InCure(DOCTOR.Doctor#),
Town(TOWN.Town))

E. Defining Attributes in Dimension Tables

All foreign keys may be dropped from the dimension tables since they are not necessary. Now the dimension tables are:

DOCTOR (Doctor#, DoctorName)
PATIENT(Patient#, Name)
WARD(Ward#, WardName)
OP-THEATRE(OpTheatre#, OpTheatreType)
DIAGNOSIS(Diagnosis#, Name)
TIME(Date, Month, Quarter, Year) with
Date→Month→Quarter→Year hierarchy
TOWN(Town, TownName, Region) with
Town→Region hierarchy
SEX(Sex, Name)
AGE(Age, Age10) with Age→Age10 hierarchy
RESULT(Result, ResultDescription)

The fact tables are:

OPERATION(Operation#, *OperationDate*(TIME.Date),
OperationType:value-per-unit, Admission#,
OpTheatre# (OP-THEATRE.OpTheatre#),
Doctor#Operator (DOCTOR.Doctor#),
AdmissionDate(TIME.Date),
Result(RESULT.Result),
Doctor#Admission(DOCTOR.Doctor#),
Patient#(PATIENT.Patient),
Ward#(WARD.Ward#),
Diagnosis#(DIAGNOSIS.Diagnosis#),
OperationWard#(WARD.Ward#),
Sex(SEX.Sex), *Age*(AGE.Age),
Doctor#InCure(DOCTOR.Doctor#),
Town(TOWN.Town))
ADMISSION(Admission#, *AdmissionDate*(TIME.Date),
NumberOfDays:level, Value:level,
Result(RESULT.Result),
Patient#(PATIENT.Patient#),
Diagnosis#(DIAGNOSIS.Diagnosis#),
Doctor#Admission(DOCTOR.Doctor#),
Ward#(WARD.Ward#),
Sex(SEX.Sex), *Age*(AGE.Age),
Doctor#InCure(DOCTOR.Doctor#),
Town(TOWN.Town))
ADMISSION-SECONDARY-DIAGNOSIS(Admission#,
Diagnosis#Secondary(DIAGNOSIS.Diagnosis#),
AdmissionDate(TIME.Date),
Result(RESULT.Result),
Doctor#Admission(DOCTOR.Doctor#),
Patient#(PATIENT.Patient#),
Ward#(WARD.Ward#),
Diagnosis#(DIAGNOSIS.Diagnosis#),
Sex(SEX.Sex), *Age*(AGE.Age),
Doctor#InCure(DOCTOR.Doctor#),
Town(TOWN.Town))

F. Checking Multidimensional Schema(s)

Expressiveness of Multidimensional Schema

The expressiveness of multidimensional schema is retained while steps 1.1b and 3.1 in **D** are not done.

Multidimensional Normal Form

We may check that every dimensional schema of the example satisfies dimensional normal form, while in each dimensional schema there exists exactly one terminal attribute and its values are complete.

Summarizability

The type of every fact attribute (flow, level or value-per-unit) is specified and declared in the multidimensional schema. Also, in the graphical notation the temporal dimension time is marked by a dot.

The resulting entity-relationship-like model is shown in Fig. 7. showing fact tables in gray. Various arrows are used as pointers from fact tables to their dimension tables or as pointers showing dimension hierarchies. Dotted arrow shows degenerate dimension.

4. Validation of the Procedure

How do we validate that the output multidimensional schema is valid? The easiest way is to validate that all primary functional dependencies of the relational schema are contained in the multidimensional schema. Primary functional dependency is an interentity functional dependency that exists among foreign key of an entitity and primary key of another entity, which is referred to as referential integrity. Primary functional dependencies implement relationships among entities of an entity-relationship model. If the mentioned condition is satisfied, the multidimensional schema produced by the procedure is at same time equally expressive as its initial relation schema.

In the multidimensional schema all primary functional dependencies result from fact tables

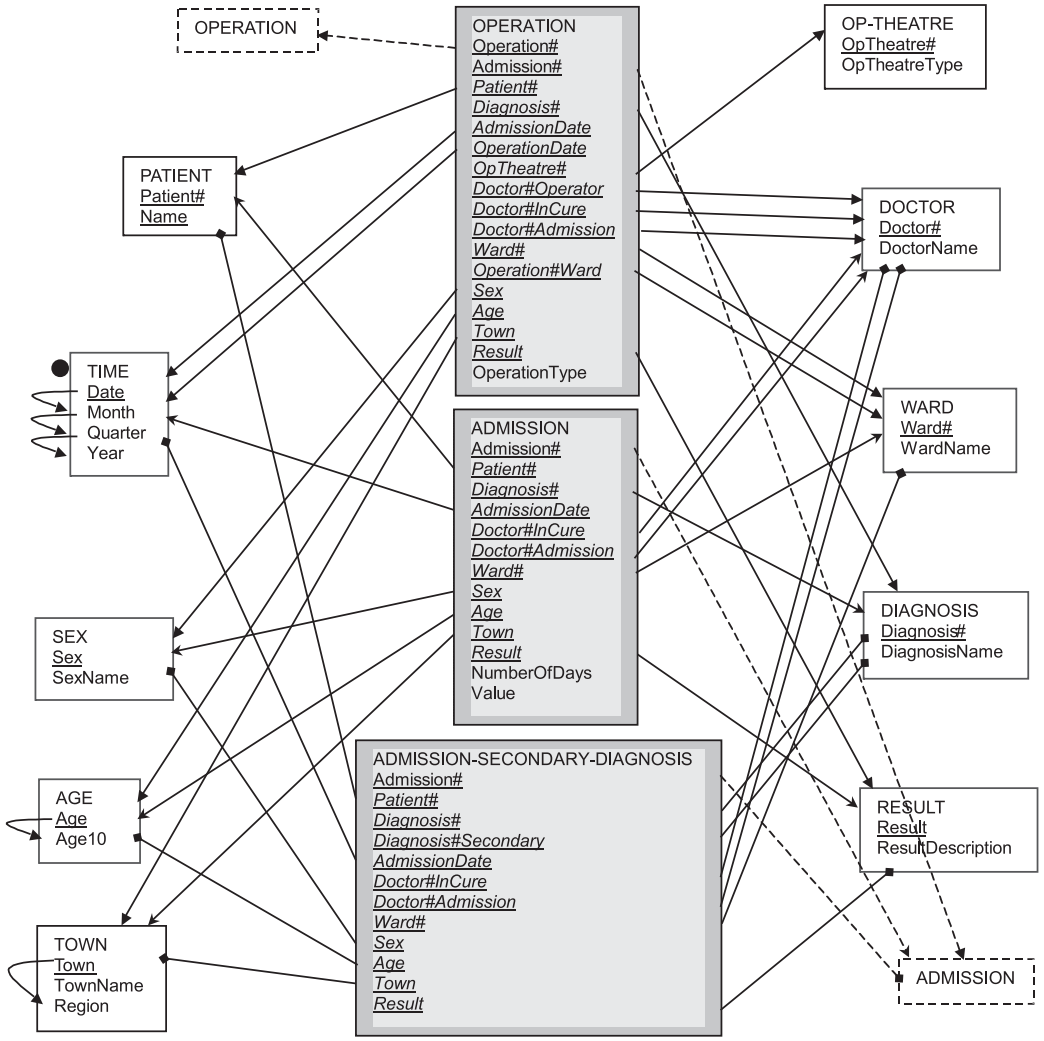


Fig. 7. Multidimensional model of the hospital example.

to dimension tables. The step **D**.1.3 of the procedure ensures by the foreign key transition that all primary functional dependencies that exist in the relational schema are transferred into functional dependencies resulting from fact tables in the multidimensional schema.

For example, in the example the following functional dependencies of foreign keys to primary keys in the relational schema

OPERATION.Admission#→ADMISSION.Admission#
ADMISSION.Patient#→PATIENT.Patient
PATIENT.Town→TOWN.Town

are substituted by direct functional dependency

OPERATION.Town→TOWN.Town

in the multidimensional schema.

5. Conclusion

In the paper, the general procedure of conversion of relational into multidimensional database schema is presented. It consists of the following steps: discovering potential dimensions, defining fact and dimension attributes, defining fact and dimension tables and defining attributes in tables. The presented procedure is semi-automated while in the course of conversion some decisions must be made.

Acknowledgement

Many thanks to the anonymous referees for their helpful and insightful comments.

References

[1] M. GOLFARELLI, S. RIZZI, *A Methodological Framework for Data Warehouse Design. Proc. 1st Int. ACM Workshop on Data Warehousing and OLAP*, (1998), Washington.

[2] M.GOLFARELLI, D. MAIO, S. RIZZI, *Conceptual Design of Data Warehouses from E/R Schemes. Proc. of the Hawaii Int. Conf. on System Science*, (1998a), Kona.

[3] M. GYSSENS, L. LAKSHMANAN, *A Foundation for Multi-Dimensional Databases. Proc. 22nd VLDB Conference*, (1996), Mumbai (Bombay).

[4] R. KIMBALL, *The Data Warehouse Toolkit. John Wiley*, (1996), New York.

[5] W. LEHNER, H. ALBRECHT, H. WEDEKIND, *Normal Forms for Multidimensional Databases. Proc. 10th Int. Conf. on Scient. and Statistical Database Management*, (1998), pp. 63-72, Capri.

[6] J. MARTIN, J. ODELL, *Object Oriented Methods*; A Foundation. Prentice-Hall, (1995), Englewood Cliffs.

[7] H-J. LENZ, A. SHOSHANI, *Summarizability in OLAP and Statistical Data Bases. 9th International Conference on Statistical and Scientific Database Management (SSDBM)*, (1997).

[8] A. SHOSHANI, *OLAP and Statistical Databases: Similarities and Differences. 16th ACK SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, (1997), pp. 185-196.

Received: November, 1999
Revised: March, 2002
Accepted: May, 2002

Contact address:
Mladen Varga
Graduate School of Economics & Business
University of Zagreb
Trg J. F. Kennedyja 6
10000 Zagreb
Croatia
Phone: +385 1 2383279
Fax: +385 1 2335633
e-mail: mladen.varga@efzg.hr

MLADEN VARGA obtained his B. Sc. in Electronics, M. Sc. and Ph. D. in Computer Science from the Faculty of Electrical Engineering and Computer Science at the University of Zagreb. Currently he is a professor at the Department of Computing at the Graduate School of Economics & Business, University of Zagreb. His main research interests are data modelling, data base systems, data warehousing systems and software engineering methods. He has published over 40 papers and several books.