International Journal of Database Theory and Application
Vol.8, No.3 (2015), pp.111-130
http://dx.doi.org/10.14257/ijdta.2015.8.3.10

Implementation of Star Schemas from ER Model

Gunjan Chandwani and Veepu Uppal

Manav Rachna College of Enginering
gunjanchindwani.mrce@mrei.ac.in, veepu.mrce@mrei.ac.in

Abstract

A Star Schema is representation of a data warehouse which is used in strategic decision making and analysis. In this paper we present a method ER Diagram is converted into a star schema.

Keywords: *Data Warehouse, StarSchema*

1. Introduction

In most database environments, users perform two basic types of tasks: modification (inserting, updating, and deleting records) and retrieval (queries). Modifying records is generally known as online transaction processing (OLTP). Data retrieval is referred to as online analytical processing (OLAP) or decision support, because the information is often used to make business decisions. This section describes these data models and their structural requirements.

When database records are modified, the most important requirements are update performance and data integrity. These needs are addressed by the entity relation model of organizing data. Entity relation schemas are highly normalized. This means that data redundancy is eliminated by separating the data into multiple tables. The process of normalization results in a complex schema with many tables and joins paths.

When database records are retrieved, the most important requirements are query performance and schema simplicity. These needs are best addressed by the dimensional model. Another name for the dimensional model is the star schema.

.

1.1. Star Schema[1, 2]

A diagram of a star schema resembles a star, with a fact table at the center. Figure 1 is a sample star schema.

A fact table usually contains numeric measurements, and is the only type of table with multiple joins to other tables. Surrounding the fact table are dimension tables, which are related to the fact table by a single join. Dimension tables contain data that describe the different characteristics, or dimensions, of a business. Data warehouses and data marts are usually based on a star schema.

In a star schema, subjects are either facts or dimensions. You define and organize subjects according to how they are measured and whether or not they change over time. Facts change regularly, and dimensions do not change, or change very slowly.

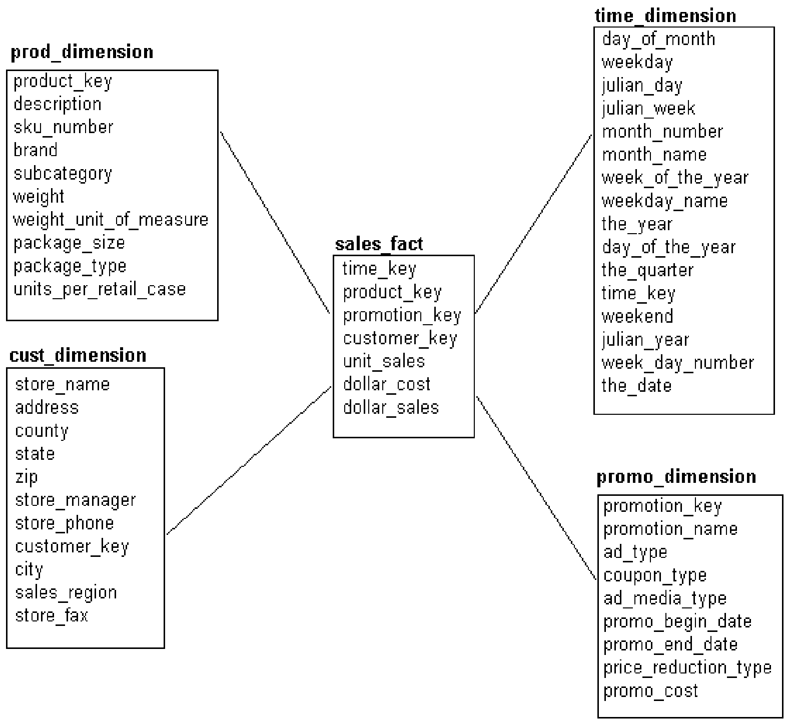


Figure 1. Star Schema

1.1.1 Advantages: Star schemas are easy for end users and applications to understand and navigate. With a well-designed schema, users can quickly analyze large, multidimensional data sets. The main advantages of star schemas in a decision-support environment are:

- **Query Performance:** Because a star schema database has a small number of tables and clear join paths, queries run faster than they do against an OLTP system. Small single-table queries, usually of dimension tables, are almost instantaneous. Large join queries that involve multiple tables take only seconds or minutes to run.In a star schema database design, the dimensions are linked only through the central fact table. When two dimension tables are used in a query, only one join path, intersecting the fact table, exists between those two tables. This design feature enforces accurate and consistent query results.

- **Load Performance and Administration:** Structural simplicity also reduces the time required to load large batches of data into a star schema database. By defining facts and dimensions and separating them into different tables, the impact of a load operation is reduced. Dimension tables can be populated once and occasionally refreshed. You can add new facts regularly and selectively by appending records to a fact table.

- **Built-in Referential Integrity:** A star schema has referential integrity built in when data is loaded. Referential integrity is enforced because each record in a dimension table has a unique primary key, and all keys in the fact tables are legitimate foreign keys drawn from the dimension tables. A record in the fact table that is not related correctly to a dimension cannot be given the correct key value to be retrieved.

- **Easily Understood:** A star schema is easy to understand and navigate, with dimensions joined only through the fact table. These joins are more significant to the end user, because they represent the fundamental relationship between parts of the

Copyright ⓒ 2015 SERSC

underlying business. Users can also browse dimension table attributes before constructing a query.

1.1.2 Disadvantages: A Star schema has a narrow scope in terms of the fact and dimensions represented as compared to the relational model.

- A star schema is good to store current data which may be historical, aggregated, detailed.
- Not suitable for storing detailed data.

2. ER Schema

This model incorporates some of the important semantic information about the real world. The entity-relationship model can be used as a basis for unification of different views of data: the network model, the relational model, and the entity set model. The entity-relationship model adopts the more natural view that the real world consists of entities and relationships. An entity is a “thing” which can be distinctly identified. A specific person, company, or event is an example of an entity. A relationship is an association among entities. For instance, “father-son “is a relationship between two “person” entities.’

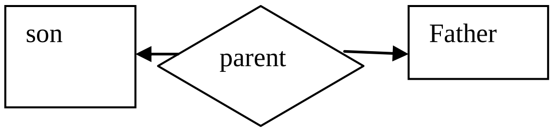


Figure 2. Simplified ER Diagram

As we have already stated that there are many approaches for conversion of an ER Schema to a Star Schema, amongst them we have chosen Moody’s and Golfarelli’s algorithm for converting an ER Schema to a Star schema.

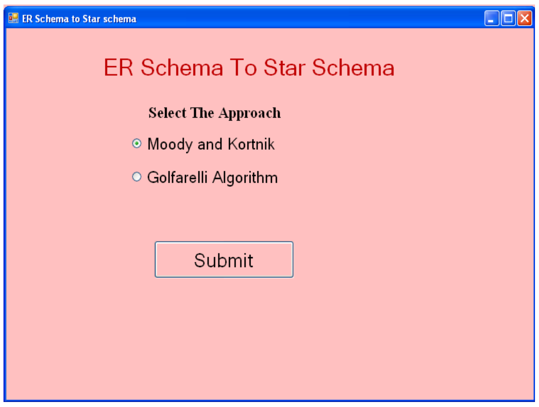


Figure 3. Select Approach

3. Moody’s and Kortnik Approach for Designing a Star Schema From an ER Schema [8, 9]

3.1. Introduction of the Algorithm:

Moody’s and Kortnik algorithm describes a method for developing dimensional models from traditional Entity Relationship models. This can be used to design data warehouses and data marts based on enterprise data models.

3.2. Principle

3.2.1 Chunking: As we know that a data warehouse contains a lot of information but humans have a limited capacity for processing information. The primary mechanism used by the human mind to cope with large amounts of information is to organize it into "chunks" of manageable sizes. The ability to recursively develop information-saturated chunks is the key to people's ability to deal with complexity every day. The process of organizing data into a set of separate star schemas effectively provides a way of organizing a large amount of data into cognitively manageable "chunks.

3.2.2 Hierarchical Structuring: Hierarchy is one of the most common ways of organizing complexity for the purposes of human understanding. Hierarchical structures act as a complexity management mechanism by reducing the number of items one has to deal with at each level of the hierarchy. Hierarchical structures are a familiar and natural way of organizing information, and are all around us in everyday life.Each dimension in a star schema typically consists of one or more hierarchies. These provide a way of classifying business events stored in the fact table, thereby reducing complexity. It is this hierarchical structure that provides the ability to analyze data at different levels of detail, and to "roll up" and "drill down" in OLAP tools. The process of building dimensional models is largely one of extracting hierarchical structures from enterprise data.

3.3 Input to the Algorithm

The input to the Algorithm is an ER Schema. Moody's from his experiments found that a star Schema is just a restricted form of an ER Schema. There is a single entity called fact table which is in 3NF (third normal forms).Violation to second normal form (2NF) would result in double counting in queries.

The fact table forms an n-array intersection entity (where n is the number of dimension) between the dimension tables, and includes keys of all dimension tables.

There are one or more entities called dimension tables each of which is related to fact table via one or more one-to- many relationships. Dimension tables have simple keys and are atleast in 2NF.

The following shows the sample ER Schema that is being transformed to a star schema:

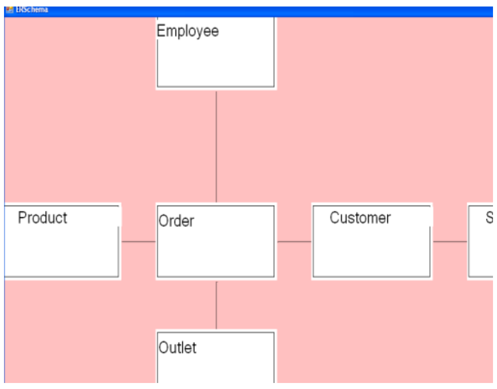


Figure 4: A Sample ER Schema Used as Input to Moody Approach

Deriving dimensional models from ER models also provides a more structured approach to dimensional design than starting from first principles. There is a large conceptual "leap" in getting from end-user analysis requirements to dimensional models.

3.4. Working of the Algorithm

The Moody's approach of transformation of ER model to a star schema is a process of selective subsetting, de- normalization and optional summarization. The Transformation of ER Schema to Star schema takes place in four steps:

Step 1: Classify Entity
Step 2: Design high level star schema
Step 3: Design detailed fact table
Step 4: Design Detailed dimension Table

3.4.1. Classify Entity: The first step in transformation approach is to classify entity into three distinct categories:

A) Transaction Entities: These entities record details of business events (*e.g.*, orders, shipments, payments, insurance claims, bank trans-actions, hotel bookings, airline reservations, and hospital admissions). Most decision support applications focus on such events to identify patterns, trends, and potential problems and opportunities. The exception to this rule is the case of" snapshot entities": entities recording a static level of some commodity at a point in time (*e.g.* account balances and inventory levels).These record effect of business on the state of an entity.

B) Component Entities: These entities are directly related to transaction entities by one – to- many relationships. These are involved in the business event and answer "who","what","when","whom","how" and "why" questions about the event.

c) Classification Entities: These entities are related to a component entity by a chain of one-to-many relationships. These define embedded hierarchies in the data model and are used to classify component hierarchies.

The classification of entities for the sample data model is: There are two transaction level entities in the model Order, Order item as they correspond to business events. They represent a different level of detail for the same business event.

There are four component Entities Product, customer, Retail outlet, Employee.

There are many classification entities defining separate but partially overlapping hierarchies in the model.Some of the entities in the above specified ER model do not fit into the hierarchical structure of the dimensional Model and hence cannot be represented in the form of a star schema. Thus the process of dimension-alizing ER Schema weeds out non hierarchial data.

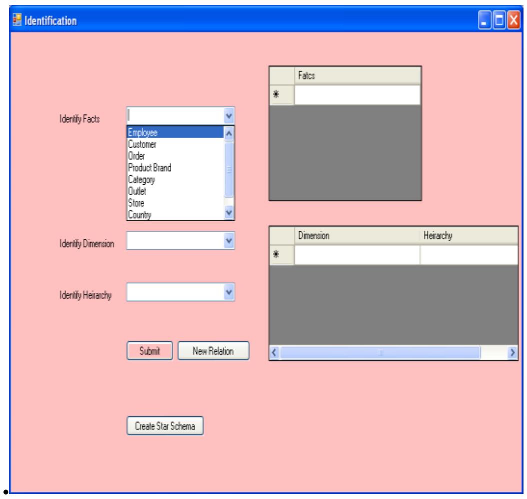
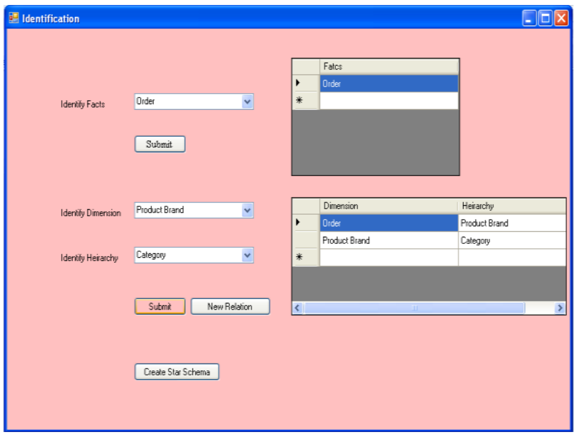


Figure 4. Identify Facts



Identify Facts: Order

	Fatcs
	Order

Identify Dimension: Product Brand

Identify Heirarchy: Category

	Dimension	Heirarchy
	Order	Product Brand
	Product Brand	Category

Figure 5. Identify Dimension and Hierarchies

3.4.2 High Level Star schema design: In this step relevant star schemas are identified and their high level structures are defined (table level design).
In this step transaction entity corresponds to fact tables and component entity corresponds to dimension tables, but mapping is not always one to one.

a) Identify star schema required: Each transaction entity is a candidate for a star schema. Each star schema is based on a single business event and hence represents a manageable sized chunk of data. There is not always a one to one correspondence between transaction entities and star schemas.

Not all transaction entities are important for decision making; user input will be required to choose relevant transactions. Multiple star schemas at levels of detail may be required for a particular transaction.

b) Define level of summarization: To design a star schema we have to choose the appropriate level of granularity-*i.e.* the level of details at which the data is stored. We have two levels of granularity:

i) Unsummarized (transaction level granularity): this is the highest level where each fact table corresponds to a single transaction.

ii) Summarized: transactions may be summarized by a subset of dimensions or dimensional attributes. In this case each row in a fact table corresponds to multiple transactions.

The lower the level of granularity *i.e.* higher the level of summarization the less storage will be required and queries will be executed much faster.Summarization looses information and limits the types of analysis that can be carried out. It is not always necessary that all star schemas should contain summarized data.Any combination of dimensions or dimensional attributes may be used to summarize transactions. For *e.g.* Order could be summarized by:
1. Month (an attribute of date dimension) and retail outlet: this gives monthly sales total for each outlet.
2. Date, Product, city (an attribute of retail outlet dimension):this gives daily product sales for each outlet.

c) Identify Relevant Dimensions: The component entity associated with each transaction entity represents candidate dimensions for the star schemas. However not every component entity is relevant for the purpose of analysis or granularity chosen.
Explicit dimension are required to represent date and time to support historical analyses.

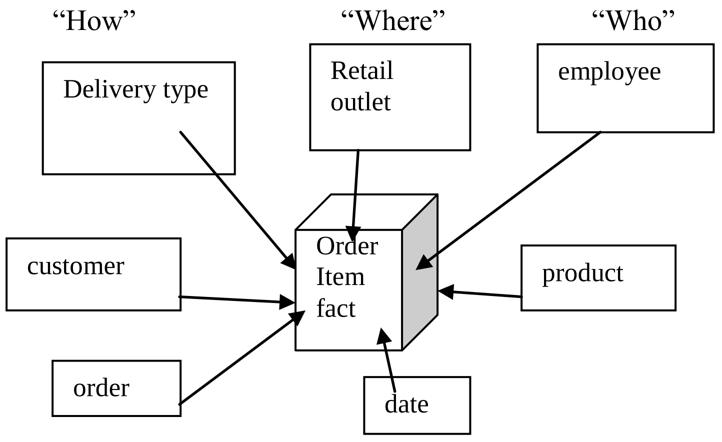


Figure 6. Transaction Level Star Schema for Order /Order Item Transaction Entities

In Figure 6 we see that a star schema has six dimensions corresponding to five component entities relating to transaction plus a date dimension.

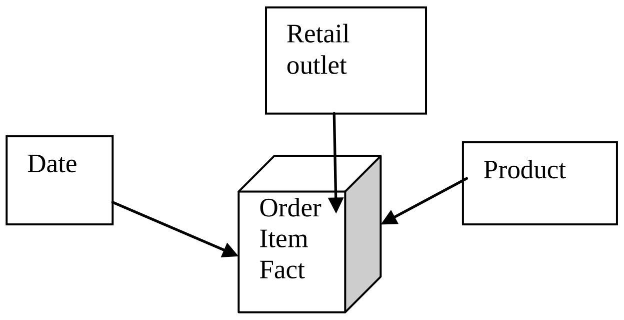


Figure 7. Summary Level Star Schema for Order Transaction

The Figure 7 shows a summary star schema for order transaction in which daily reports are summarized by Retail outlet, Date, Product. When non transaction level granularity (summary) is chosen, the dimensions required will be determined by how the transactions are summarized, this will be subset of the number of dimensions required in transaction level star schema.

3.4.3. Detailed Fact Table Design: The Design process includes following steps.

a) Define Key: The key of each fact table is a composite key consisting of all the keys of all dimension tables. This key is not minimal unlike of relational databases.

b) Define Fact: The non key attributes of the fact table are measures (fact) that can be analyzed using numerical functions. What facts are defined depends on the event information collected by operational systems-that is attributes that are stored in transaction entities.

To define fact we use additivity:

i) **Fully additive facts**: These are the facts that can be meaning fully added across all dimensions. For example Qty ordered in the order item entity can be added across dates, product, customer to get total sale volume for particular day, product, and customer.
ii) **Semi additive facts:** these are the facts that can be meaning fully added to some dimensions but not all. For example Qty on hand can be averaged over time.
ii) **Non Additive facts**: These are the facts that cannot be meaning fully added across any dimension. For *e.g.* Unit price from order item entity cannot be added across any dimension

In Figure 6 each row in the fact table corresponds to an individual order item. The key of the fact table consists of the keys of all dimension tables.
The Figure 7 shows the detailed fact table for summary level star schema. Each row in the fact table summarizes sales by date, product, and retail outlet.

3.4.4. Detailed Dimension Table Design: In this step we complete the detailed design of dimension tables in each star schema. This completes the detailed design of a star schema.

a) Define Dimensional Key: The key of dimension table should be simple numeric key. Sometimes this key is the key of the underlying component entity. But we have to sometime generate operational key to keep it unique as this may cause problems while performing historical analysis.

b) Collapse hierarchies: Dimension tables are formed by collapsing or de normalizing hierarchies (defining by classification entities) into component entities. The Figure2.8 shows how the hierarchies associated with the retail outlet component are collapsed to form the Outlet dimension table. The resulting dimension table consists of union of all attributes in the original entities. It is possible for a dimension table to contain hundred of attributes. This process introduces redundancy in form of transitive dependencies which are violations to third normal form (3NF). This means that resulting dimension tables is in second normal form.

3.5 Output of the Algorithm

The Figure 8 shows the complete star schema design for order transaction:
All of the tables in the star schema are de -normalized to at least some extent: each table corresponds to multiple entities in the original normalized ER model. The dimension tables are result of collapsing classification entities into component entities.All of the dimension tables are in 2NF, they have simple keys and no repeating attributes. The fact table is in 3NF.
The date dimension is a new table which does not corresponds to any entity in the original ER model. This is because dates must be explicitly modeled in a star schema, whereas at the operational level they are represented as the data types.

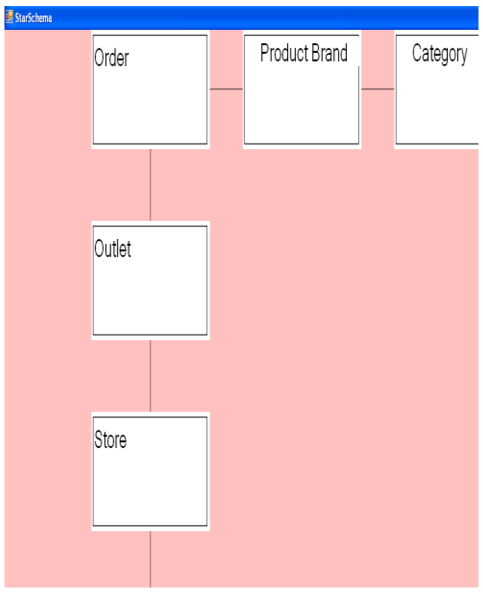


Figure 8. A Star Schema Created from ER using Moody's Approach

3.6. Conclusion

It has been derived from Moody's and Kortnik algorithm that a dimension model is just a restricted form of an ER schema and there is a straightforward transformation between the two.An ER model can be transformed into a set of dimension models by a process of selective subsetting, de normalization, and (optional) summarization.

i) Subsetting: The ER model is partitioned into a set of separate star schemas each centered on a single business event. This reduces complexity through a process called "chunking".

ii) De normalization: Hierarchies in the ER model are collapsed to form dimension tables. This further reduces complexity by reducing the number of separate tables.

ii) Summarization: the most flexible dimensional structure is one in which each fact represents a single transaction. However summarizations may be required for performance reasons, or to suit the requirements of a particular group of users.

At the detailed design level a range of transformations are required:

i) Generalizations of operational keys to ensure uniqueness of keys over time.
ii) Conversion of non additive to additive facts.

4. Golfarelli and Rizzi Algorithm [5, 7]

The Moody's Algorithm Ignores the relationship of the ER schema Golfarelli's algorithm takes into account relationships of ER Schema.

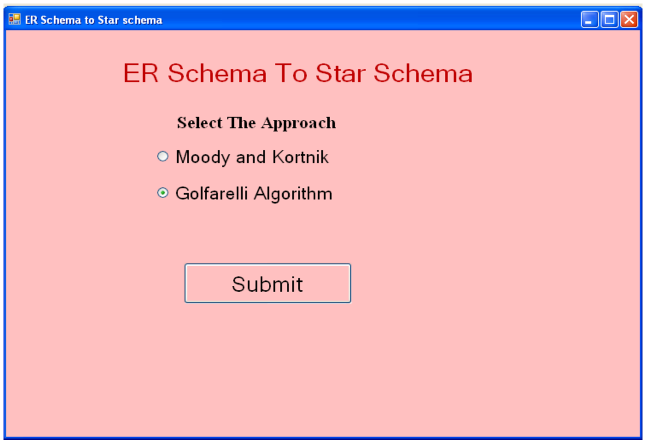


Figure 9. Select the Approach

4.1. Introduction of the Algorithm

This algorithm represents a graphical conceptual model for DWH , called Dimensional Fact Model(DFM).The representation of reality built using the DFM is called the dimensional scheme and consists of fact scheme whose basic elements are facts,dimension,hierarchies .Compatible fact scheme may be overlapped in order to relate and compare data.

4.2 Keywords Used in the Algorithm:

4.2.1. DFM: It is a dimensional scheme which consists of set of fact scheme. The components of fact scheme are facts, measures, dimension, and hierarchies.

- A fact is focus for interest for decision making process; typically it models an event occurring in the enterprise world (*e.g.* sales and shipments).
- Measures are continuously valued numeric attributes which describe the fact from different point of view; for instance each sale is measured by its revenue.
- Dimensions are discrete attributes which determine the minimum granularity adopted to represent facts; typical dimension for the sales fact are product, store, date.
- Hierarchies are made up of discrete attributes linked by –to one relationships and determine how facts may be aggregated and selected significantly for the decision making process.
- The dimension in which its hierarchy is rooted defines its finest aggregation granularity, the other dimension define coarser granularities.
- For *e.g.* hierarchy on the product dimension may include the dimension attributes product type, category, department.
- Hierarchies may also include non dimension attributes which consist of additional information about a dimensional attribute of the hierarchy and is connected by a –to many relationship, it cannot be used for aggregation.

4.2.2. Representation of Fact Scheme: In the DFM, a fact scheme is structured as a quasi tree whose root is a fact. A fact is represented by a box which reports the fact name and typically one or more measures. In the sale scheme, qty sold, revenue and no. of customers are measures.Dimension attributes are represented by circles. Each dimension attributes directly attached to the fact is a dimension. The dimension pattern of the sale scheme is {date, product, store, promotion}.Non dimension attributes are terminal within quasi tree and are represented by lines for *e.g.* addresses.Subtrees rooted in dimension are hierarchies. The arc connecting two attributes represent many– to one relationship.

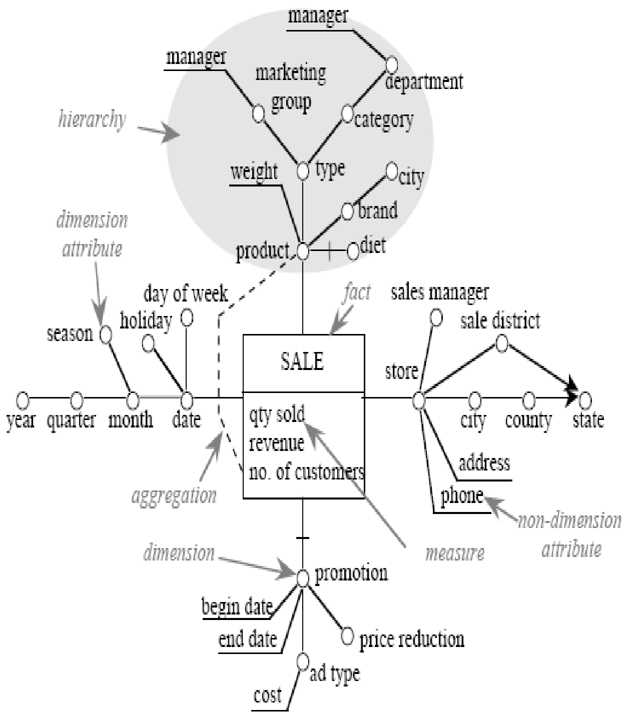


Figure 10. A Sale Fact Scheme

The fact scheme may not be a tree : in fact two or more distinct paths may connect two given dimension attributes within a hierarchy ,provided that every directed path still represents a one- to one relationship.
Optional relationship is represented by marking with a dash the corresponding arc. For example attribute diet takes value only for food products, for others it will take null value.

A measure is additive on a dimension if its values can be aggregated along the corresponding hierarchy by the sum operator.

4.2.3. Additivity: Aggregation requires defining a proper operator to compose the measure values characterizing primary fact instances into measure values characterizing each secondary fact instances.

An example of fact scheme in the example above is qty sold; the qty sold for a given sales manager is the sum of the quantities sold for all stores managed by that manager.A measure may be non additive on one or more dimension. *E.g.* are inventory levels or temperature *etc*.An inventory level is non additive on time. A temperature is non additive on all dimensions.

The Figure 11 shows an example where AVG,MIN can be used for aggregation.

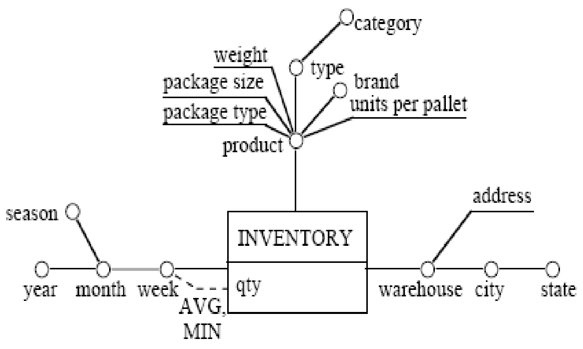


Figure 11. Inventory Fact Scheme

4.2.4. Overlapping Fact schemes: In the DFM, different facts are represented in different fact schemes. Overlapping fact schemes means combining two related fact schemes into one fact scheme if the compatibility is strict *i.e.* the inter attribute dependencies in the two schemes are not conflicting

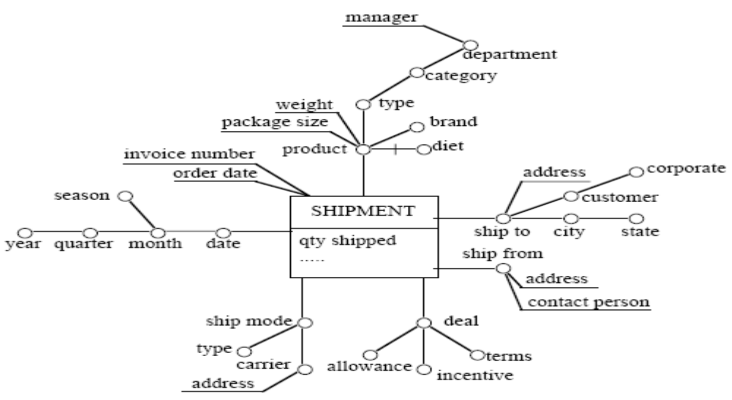


Figure 14. The Shipment Fact Scheme

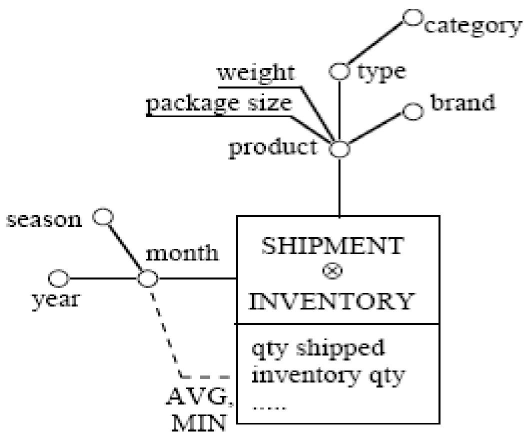


Figure 12. Overlapped Fact Scheme

The measure in resulting overlapped fact scheme is the union of the two fact schemes which are overlapped. Each hierarchy in resulting scheme includes all and only the attributes included in the corresponding hierarchies of both the fact scheme.

Copyright ⓒ 2015 SERSC

4.3 Working of the Algorithm

The transformation of ER schema to a DFM requires following steps:

Step 1: Defining Facts
Step 2: For each fact:
- a) Building the attribute tree
- b) Pruning and grafting the attribute tree
- c) Defining Dimension
- d) Defining measures
- e) Defining hierarchies

The Figure 13 shows the Simplified ER Scheme that is to be converted into DFM.

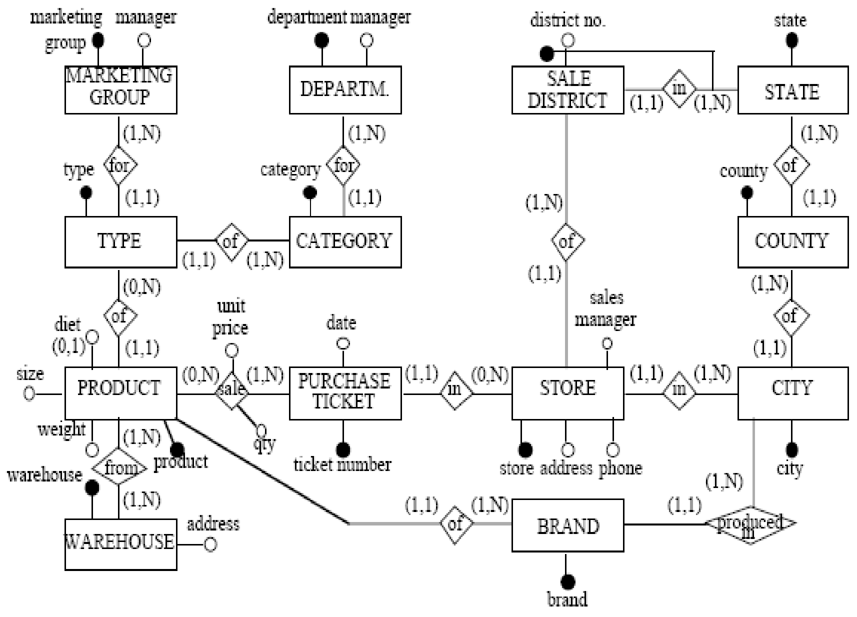


Figure 13. Simplified ER Scheme

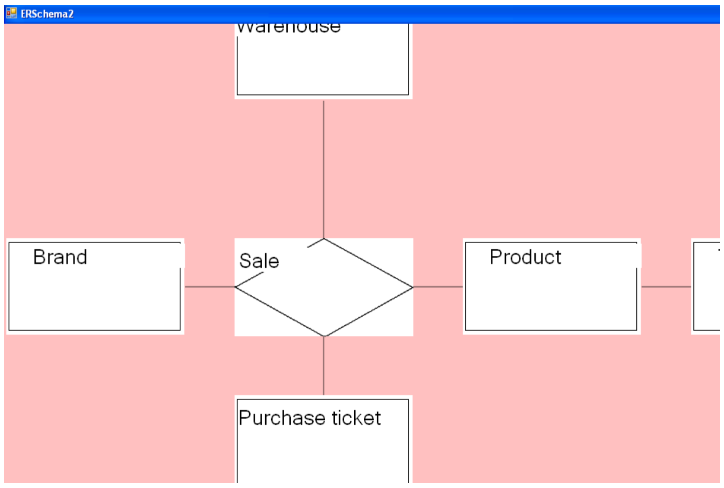


Figure 14. ER Diagram used for Conversion

4.3.1. Defining Fact: Facts are concept of primary interest for decision making. They correspond to events occurring dynamically in the world.A fact may be represented either by an entity F or by an n-array relationship R between entities E1-En.When a relationship R is a fact we have to transform this R into an entity F by replacing each branch Ei with a binary
Relationship Ri between F and Ei. The attributes of the relationship become attributes of F; the identifier of F is the combination of the identifiers of Ei.Each fact identified on the source scheme becomes the root of different fact scheme.

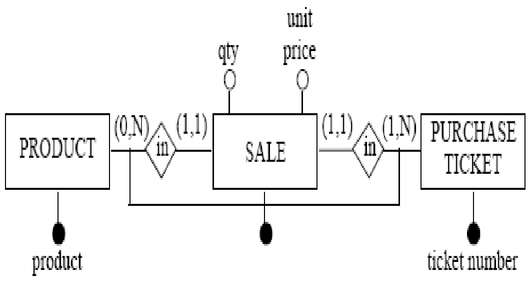


Figure 15. Transformation of Relationship Into Entity

In the above example, the fact of primary interest for business analysis is the sale of product, represented in the ER scheme by relationship sale.

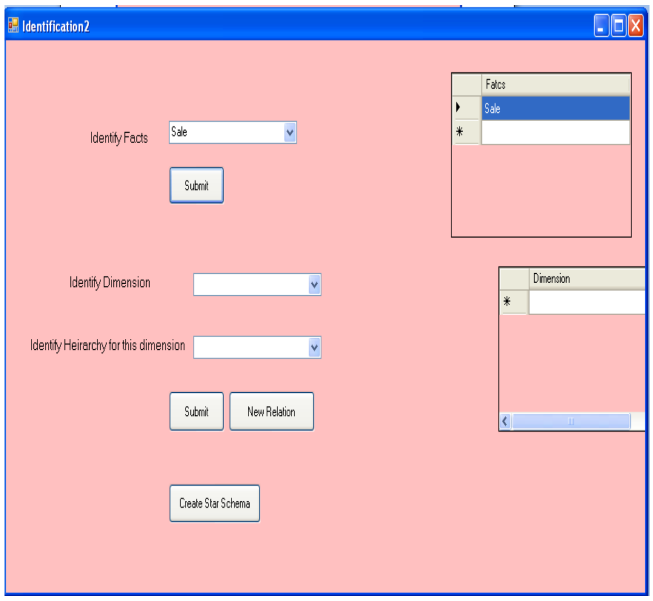


Figure 16. Define Facts

4.3.2. Building the Attribute Tree: Given a source scheme and an entity F belonging to it, we call attribute tree the quasi- tree such that:

- Each vertex corresponds to an attribute-simple or compound of the scheme.
- The root corresponds to the identifier of entity F
- For each vertex v, the corresponding attribute functionally determines all the attributes corresponding to the descendants of v.

Let identifier (E) denote the set of attributes which make up the identifier of entity E. The attribute tree for F may be constructed automatically by applying following recursive procedure.

```
root= newVertex(identifier(F));

translate (E,v):

E is the current entity, v is the current vertex
{
 for each attribute aÎE | a identifier(E) do
    addChild(v,newVertex({a})); // adds child a to vertex v
   for each entity G connected to E by a relationship R | max(E,R)=1 do
          {
            For each attribute bÎR do
                  addChild (v,newVertex({b}));
                  next=newVertex(identifier(G));
                  addChild (v,next);
                  translate (G,next);
                              }
                  }
```

In the following we illustrate how procedure translate works by showing in a step by step fashion how a branch of the attribute tree is generated.

```
root=newVertex(ticketNumber+product)
translate(E=SALE,v=sale):
      addchild(v,qty);
      addchild(v,unitPrice);
        For G=PURCHASE TICKET:
                addchild(v,ticketNumber);
                translate(PURCHASE TICKET,ticketNumber);
        for G=PRODUCT:
                addchild(v,product); translate(PRODUCT,product);
        translate(E=PURCHASE TICKET,v=ticketNumber):
        addchild(v,date);

for G=STORE:
        addchild(v,store); translate(STORE,store);
        translate(E=STORE,v=store):
        addchild(v,address); addchild(v,phone);
        addchild(v,salesManager);

for G=SALE DISTRICT:
        addchild(v,districtNo+state);
        translate(SALE DISTRICT,districtNo+state);

for G=CITY:
        addchild(v,city);
         translate(CITY,city);

translate(E=SALE DISTRICT,v=districtNo+state):
```

```
addchild(v,districtNo);
for G=STATE:
addchild(v,state); translate(STATE,state);
translate(E=STATE,v=state):
```

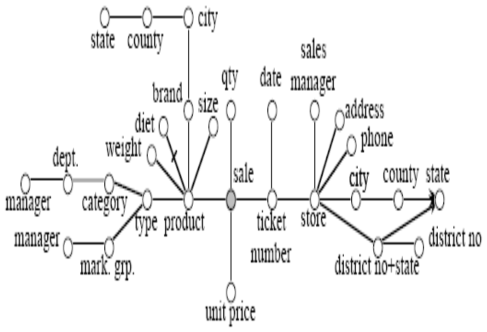


Figure 17. Attribute Tree for the Sale Example

As the attribute tree undergoes the next step in the methodology, the granularity of fact instances may change and become coarser than that expressed by the identifier of F.

Thus, in order to avoid confusion, we prefer to label the root of the attribute tree with the name of entity F rather than with its identifier. Generalization hierarchies in the E/R scheme are equivalent to one-to-one relationships between the super-entity and each sub-entity, and should be treated as such by the algorithm.

4.3.3 Pruning and Grafting the Attribute Tree: It may happen that not all of the attributes represented in the attribute tree are interesting for the DW. Thus, the attribute tree may be pruned and grafted in order to eliminate the unnecessary levels of detail.Pruning is carried out by dropping any subtree from the quasi-tree. The attributes dropped will not be included in the fact scheme, hence it will be impossible to use them to aggregate data. For instance, on the sale example, the subtree rooted in county may be dropped from the brand branch.Grafting is used when, though a vertex of the quasi-tree expresses an uninteresting piece of information, its descendants must be preserved; for instance, one may want to classify products directly by category, without considering the information on their type.

Let v be the vertex to be eliminated:

```
graft(v):
{ for each v' | v' is father of v do
        {for each v'' | v'' is child of v do
                {addChild(v',v'');
                        drop v;
                }
        }
}
```

Thus, grafting is carried out by moving the entire subtree with root in v to its father(s) v';if we denote with t the attribute tree and with I the set of its vertices, procedure graft(v) returns cnt(t,I-{v}). As a result, attribute v will not be included in the fact scheme and the corresponding aggregation level will be lost; However on the other hand, all the descendant levels will be maintained. In the sale example, the detail of purchase tickets is uninteresting and vertex ticket number can be grafted. In general, grafting a child of the root corresponds to making the granularity of fact instances coarser and, if the node grafted has two or more children, leads to increasing the number of dimensions in the fact scheme.

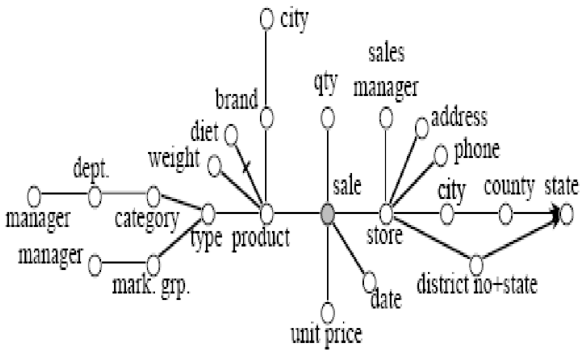
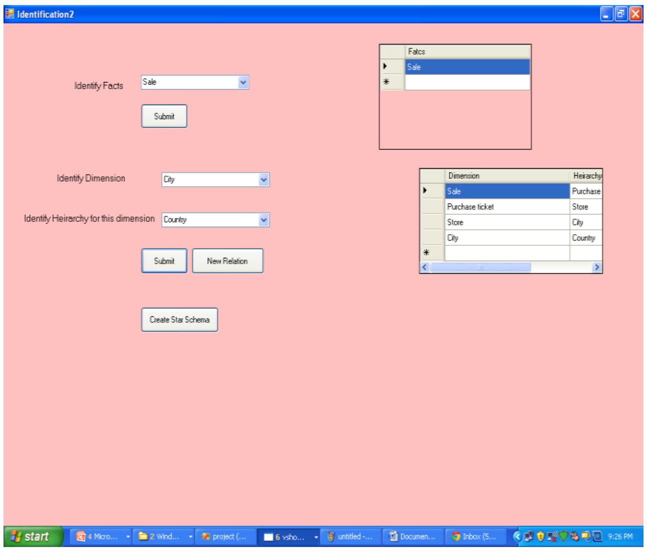


Figure 18. Attribute Tree for the Sale Example after Grafting and Pruning

4.3.4. Defining Measures: Measures are defined by applying, to numerical attributes of the attribute tree, aggregation functions which operate on all the instances (tuples) of F corresponding to each primary fact instance. The aggregation function typically consists either of the sum/average/maximum/ minimum of expressions or of the count of the number of entity instances (tuples). A fact may have no attributes, if the only information to be recorded is the occurrence of the fact. The measures determined, if any, are reported on the fact scheme.



Identify Facts: Sale

Fatcs
Sale

Identify Dimension: City

Identify Heirarchy for this dimension: Country

Dimension	Heirarchy
Sale	Purchase
Purchase ticket	Store
Store	City
City	Country

Figure 19. Identifying Dimensions

4.3.5. Defining hierarchies: Along each hierarchy, attributes must be arranged into a quasi-tree such that a one-to-one relationship holds between each node and its descendants. The attribute tree already shows a plausible organization for hierarchies; at this stage, it is still possible to prune and graft the quasi-tree in order to eliminate irrelevant details.

It is also possible to add new levels of aggregation by defining range for numerical attributes. Typically, this is done on the time dimension. In the sale example, the time dimension is enriched by introducing attributes month, quarter, *etc*.

During this phase, the attributes which should not be used for aggregation but only for informative purposes may be identified as non-dimension attributes (for instance, address, weight, *etc*.). It should be noted that non-numerical attributes which are children of the root but have not been chosen as dimensions must necessarily either be grafted (if the granularity of the primary fact instances is coarser than that of the fact) or be represented as non-dimension (if the two granularities are equal).

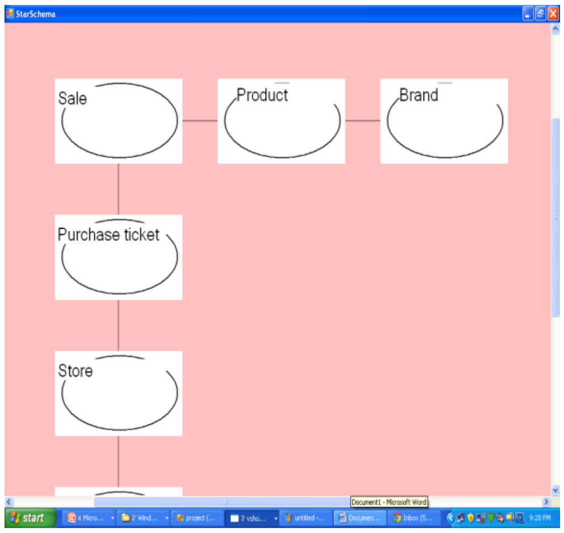


Figure 20. Star Schema Created from Golfarelli's Approach

5. Comparison of the Algorithm

From our detailed study of both the algorithms we have drawn following comparisons. They are discussed under different headings as follows:

5.1. Principle

The basic principle behind Moody's algorithm is chunking in which large amount of information is organized into small chunks of manageable sizes, and hierarchical structuring in which we reduce the number of items at each level of hierarchy., Where as the Golfarelli's design principle was to design a dimensional fact model.
Moody claims that an ER schema is just a restricted form of a star schema.

5.2. Working of the Algorithm:

The Moody's algorithm start by classifying entities into transaction, component and classification entities after which we design a high level star schema in which transaction entities corresponds to the fact (at this point a user input is required to choose which transaction entity will be relevant for decision making purposes) component entities correspond to dimensions and classification entities correspond to hierarchies. There after we design detailed fact table and detailed dimension table.The Golfarelli's algorithm starts by building a DFM by defining facts and for each fact it build a attribute tree, pruning and grafting the attribute tree, and then defining measures, hierarchies and dimensions. The basic difference between the two algorithms is that Moody's algorithm allows only entities to become facts in a star schema where as Golfarelli's algorithm allows relationship in a ER model to become facts in a DFM for this we have to convert this relationship to an entity.

Another difference is that the pruning and grafting of the attributes in DFM is done after designing of the attribute tree where as in Moody's algorithm this is done at the stage of designing a high level star schema which requires a user input to decide which transaction entities will become facts and which component entity will correspond to dimension.

The major difference is that in Moody's algorithm a star schema is represented in the form of relations (*i.e.* Facts are represented in form of fact table and hierarchies are collapsed to dimension

Which are represented in the form of a dimension table) where as in Golfarelli's algorithm a star schema is represented in the form of a quasi tree whose root is a fact and nodes directly attached to the fact are the dimensions and sub trees rooted in dimension are the hierarchies

Acknowledgement

I would like to gratefully and sincerely thank Dr. Naveen Prakash for his guidance, understanding, patience, and most importantly, his friendship during my postgraduate studies. His mentorship was paramount in providing a well rounded experience consistent my long-term career goals. He encouraged me to not only grow as an experimentalist and a chemist but also as an instructor and an independent thinker. I am not sure many graduate students are given the opportunity to develop their own individuality and self-sufficiency by being allowed to work with such independence. For everything you've done for me. I would also like to thank Ms. Veepu for helping me.

References

[1] R. Thareja, "Data warehousing", Oxford, **(2009)**.
[2] W.H. Inmon, "Building the Data Warehouse", John Wiley, New York, **(2000)**.
[3] F. Rilson and J. Freire, "DWARF: AN Approach for Requirements Definition and Management of Data Warehouse Systems", Proceeding of the 11th IEEE International Requirements Engineering Conference 1090-9, **(2003)**.
[4] B. Husemann, J. Lechtenborger, G. Vossen, "Conceptual Data Warehouse Design" Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000), **(2000)**; Stockholm, Sweden.
[5] M. Golfarelli, D. Maio and S. Rizzi, "Conceptual design of data warehouses from E/R schemes", Proc. Hawaii International Conference on System Sciences, **(1998)**; Kona, Hawaii.
[8] D. Dori, R. Feldman, A. Sturnm, "Transforming an Operational System Model to a Data Warehouse Model: A Survey of Techniques", Proceedings of the IEEE International Conference on Software - Science, Technology & Engineering
[9] M. Golfarelli, S. Rizzi, "Designing the Data Warehouse: Key Steps and Crucial Issues", Journal of Computer Science and Information Management, vol. 2. no.3, **(1999)**.
[10] N. Prakash, A. Gosain, "An approach to engineering requirement of datawarehouses, Requirement enginnering", vol 1, no.13, **(2008)**, pp. 49-72.
[11] D.L. Moody and M.A.R. Kortink, "From ER Models to Dimensional Models: Bridging theGapbetween OLTP and OLAP Design", Journal of Business Intelligence, vol. 8, **(2003)**.
[12] D.L. Moody and M.A.R**.** Kortink**,"** From ER Models to Dimensional Models: Advanced Design Issues", Journal of Business Intelligence, vol. 8, **(2003)**.
[13] P.P. Chen, "The entity relationship model :toward a unified view of data", ACM transactions on database system, vol. 1, no. 1,**(1976)**, pp. 9-37.
[12] www.xml.com
[13] www.xmlvalidation.com
[14] www.xmlmaster.org/en/article/d01/c04

Authors

Gunjan Chandwani
Assistant Professor
Manav Rachna College of Enginering

Veepu Uppal
Assistant Professor
Manav Rachna College of Enginering

Copyright ⓒ 2015 SERSC