Birzeit University
FACULTY OF INFORMATION TECHNOLOGY
Scientific Computing Master Program

USING DATABASE THEORIES IN SCIENTIFIC APPLICATIONS

By

Saleem Ahmad Jayousi

Supervisors

Dr. Hassan Shibly

Dr. Samir Matar

August, 2007
Birzeit, Palestine

Table of Contents

Abstract

Traditional methods used for describing scientific methods such as manuals must be replaced. Those old methods consume a lot of time in designing and analyzing.

Any machine were bought to home, office or company should be supported by electronic catalog, this catalog depends on database management. So technician and engineer refers to this programmable catalog to find errors or develop the system.

Scientific systems described by data, so database models must be used to manage those data. Actually using data management theories and methods make it easy to understand the system, detect its errors, develop it and trace it.

Each database method can serve managing scientific systems in a different matter, so each one has its advantages and disadvantages.

Using data models for scientific applications give the following benefits:

1- Make it easy to understand models.

2- When manipulating these models by computers, make it easy to answer requests and get results quickly.

3- Gives accurate answers for user queries.

Hierarchal data model draws the systems exactly, make it easy in detecting system errors. And ease the system improvements. It may take a time in system design specially in big systems. It also limits some relations between system functional units and parts.

Entity Relationship (ER) Data Model seems to be the most suitable model. ER model can represent entities and the relationship between them completely. All functional links can be represented in ER model. ER model allows recursive links.

Relational model suffers because it can't represent relationships obviously. Relational model disallow recursive links.

Network Model represents entities and relationships. Many to many relationships can't be represented directly in Network Models. Recursive links disallowed in this model.

Introduction

Over the past several years, there has been an explosion in the amount of information available to both business and scientific enterprises. Individuals and corporations routinely use computers both to record proprietary information and to distribute public information via the WWW.

The challenge facing data managers today is how to fully utilize this wealth of information without overwhelming either the end user or the system maintainers. Because of the competitive advantage provided by better data analysis, business information processing has driven technological advances in data warehousing and analytic processing. Within this community, relational databases are accepted as standards for transaction-based systems and SQL provides a consistent, well-known data access and manipulation language. Several commercial tools, from companies such as Red Brick, Cerebellum, Sugent, Informatics, VIT, and many others, address the steps involved in creating, maintaining, and analyzing a warehouse.

Unfortunately, scientific applications face unique problems that are not being addressed by those tools. While part of the

problem arises from the lack of standardization in scientific domains for example, information sources do not share a common terminology, data representation, or data management architecture the primary problems are the subtle but complex relationships between data and the dynamic source schemata [18].

The inability to fully utilize the wealth of publicly available information is a significant problem for scientific domains in general. While commercial products are currently available, they are focused on business applications and do not meet the unique needs of scientific domains. In particular, they do not address either the subtle data integration issues resulting from the complex relationships between scientific data, or the more obvious schema integration issues resulting from the dynamic nature of the sources.

The scientific applications contain a lot of elements, such as chemical, mechanical or electronic parts. Manuals usually have block diagrams that explain these applications which always hard to understand and trace system through it.

Computation the Scientific applications data seems to be very important. It is the best replacement for catalogs and diagrams.

Computing any system stands basically on application database. Scientific applications database are manipulated to be partitioned to data modules, and then these data can be computed, so it become easy to be analyzed and understood.

The problem is how to qualify applications data to be data based and which database theory serves more scientific applications. The most important theories in database are hierarchal, Relational, Entity Relationship, Network theories. Properties of each one, and which serves scientific applications more will be discussed.

Data warehouses and data marts have been successfully applied to a multitude of commercial business applications. They have proven to be invaluable tools by integrating information from distributed, heterogeneous sources and summarizing this data for use throughout the enterprise. Although the need for information dissemination is as vital in science as in business, working warehouses in this community are scarce because traditional warehousing techniques don't transfer to scientific environments.

There are two primary reasons for this difficulty. First, schema integration is more difficult for scientific databases than for business sources, because of the complexity of the concepts and the

associated relationships. While this difference has not yet been fully explored, it is an important consideration when determining how to integrate autonomous sources. Second, scientific data sources have highly dynamic data representations (schemata). When a data source participating in a warehouse changes its schema, both the mediator transferring data to the warehouse and the warehouse itself need to be updated to reflect these modifications. The cost of repeatedly performing these updates in a traditional warehouse, as is required in a dynamic environment, is prohibitive [18].

Till this moment Database serves sincerely management and financial systems. In this thesis the validity of serving database for scientific systems will be checked.

The main objective in this thesis is to "***Setup main outlines for a new mechanism called Database Modeling for Scientific Systems (DMSS), which will serve experts in analyzing scientific systems***".

The following table illustrates some applications which served by database systems.

Application	Database Theories Support	Software Support

Hospital patients Doctors Rooms departments	Patients Table ## ## / ## ## / ## ## Doctors Table ## ## / ## ## / ## ## Departments Table ## ## / ## ## / ## ## Relational, Entity Relationship, Network Theories **Can** Support	**Oracle, SQL, Access software** **can** **support.**
Company Employees Stores Sales departments	Employees Table ## ## / ## ## / ## ## Stores Table ## ## / ## ## / ## ## Sales Table ## ## / ## ## / ## ## Relational, Entity Relationship, Network Theories **Can** Support	**Oracle, SQL, Access software** **can** **support.**
Electrical Control System Motors Sensors Power Supplies Emmiters	**?**	**?**

First chapter will discuss the traditional methods used to explain and analyze scientific systems, its difficulties and disadvantages. And the imperative degree of scientific systems for computing method will be discussed.

The second chapter tries to start drawing the outlines of computing scientific applications. Thesis will discuss how to manipulate and manage scientific applications data to be qualified for data basing then computing.

Third chapter explains the meaning of database, database model and what the main factors in scientific database design are.

Up to The fourth chapter, applying the new mechanism (DMSS) in scientific systems will start. The most famous theories in database will be used to serve scientific applications. Then a comparison between database theories will be made; this comparison will be from scientific systems viewpoint.

Chapter 1

Scientific Applications

1.1 Introduction

Scientific application is any system contains modules and elements work in a tidy way to achieve a productive process. For example car, computer, plane, and washing machine are scientific applications.

Scientific applications usually designed by Specialists and engineers. When the end user starts using the application it may need to be maintained, developed, or analyzed. So designers try hardly to provide catalogs and diagrams with machines to make it easy to understand its operation.

1.2 Classical Methods

Scientific systems designers provide classical methods with machines to help engineers in maintenance procedures, those methods are:

A- Manuals: it is a documentation comes with machine, usually includes a descriptive information or illustrations about the machine, its internal structure and how does it work, etc. for example the following power supply has a catalog from TDK company:

Product	Catalog
Power Supply AC Input	

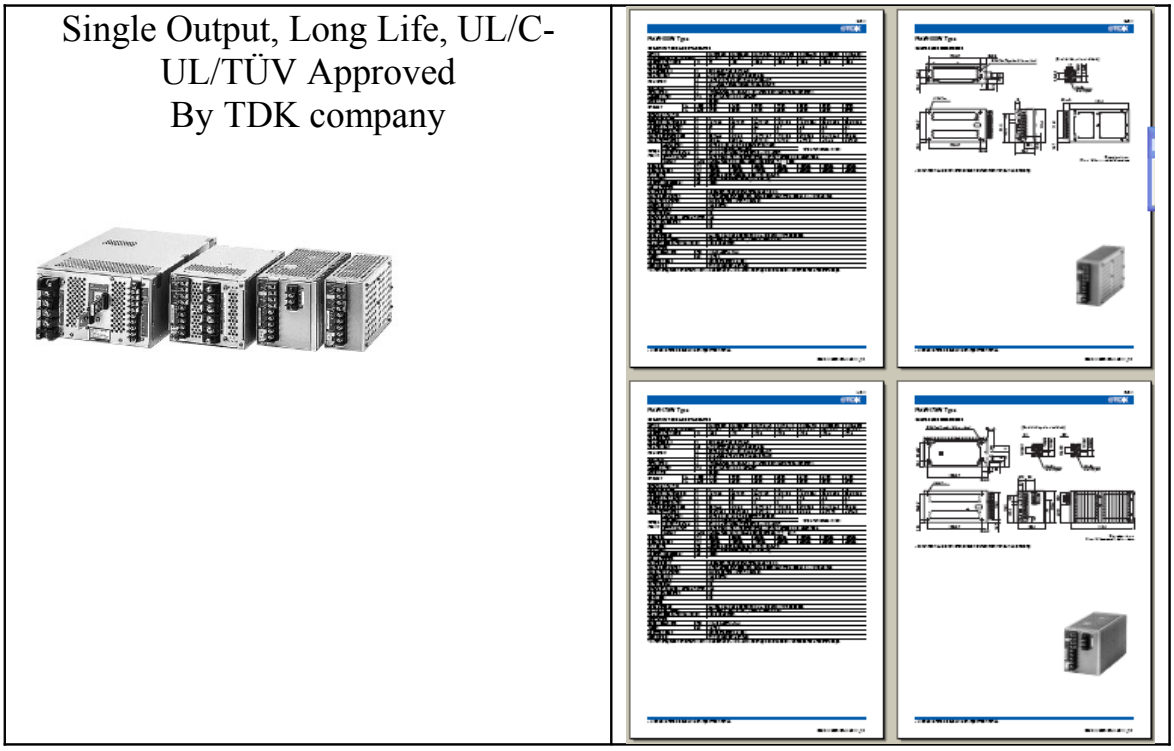


Fig 1-1 catalog as a traditional illustration method

B- Illustrative Diagrams: it is a drawn diagram or figure contains the parts of the system. It can be a block diagram consists main modules of the system, or a detailed map covers all parts in the system. The following figure explains a Diagram which comes with Hybrid Engine Vehicle of a car:

Product	Block Diagram
Car	

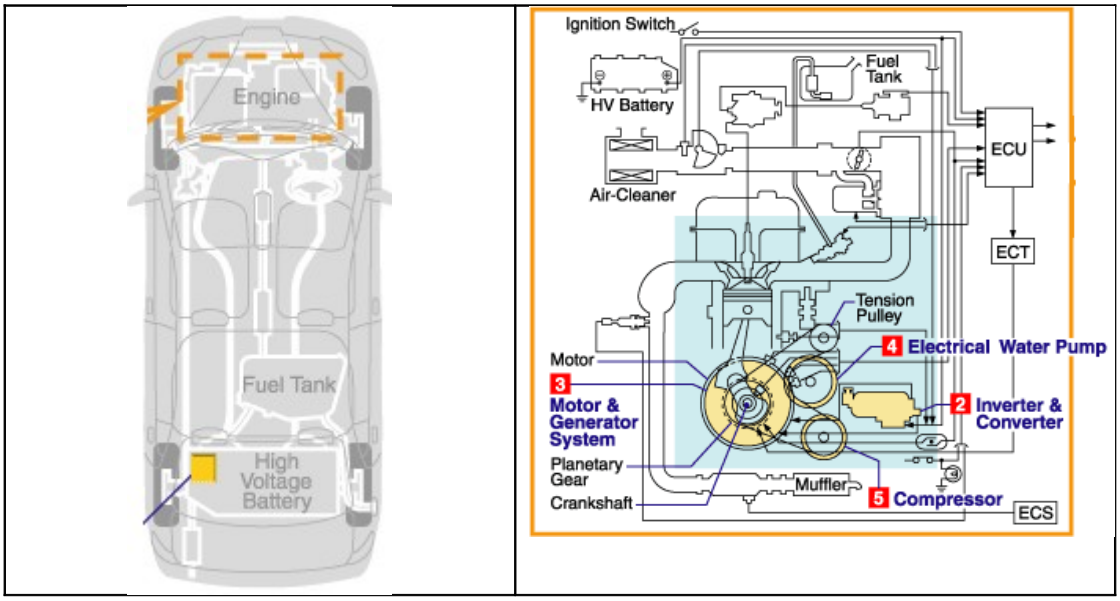


Fig 1-2 Car hybrid engine vehicle block diagram

The disadvantages of the above methods are:

1- It is difficult for experts to trace the product parts and relationships.

2- Maintenance is hard because of the complex relations.

3- It is hard to develop the product, because modules are not obvious.

Example 1.1: an expert needs a power supply for some application. He obtained a TDK power supply catalog. TDK power supply catalog (Appendix B) which illustrates the characteristics and specifications for the power supply family products (more than one type).

The expert wants a power supply with some limitations to be suitable with the application. The limitations and conditions which application needs are:

1- Power Supply must afford heat temperature 70˚C at least.

2- Power supply must provide current up to 60 A.

3- Power supply must have Current Balance (CB) Terminal (T6). CB terminal is used when several power supplies are connected in parallel to connect the respective CB terminals.

4- The shock pulse duration of the power Supply must be in range (6-16) ms.

TDK power supply catalog gives full information about 4 types of power supply in the same family. These types are RAW 100W, RAW 175W, RAW 350W and RAW 1.5KW.

Catalog in appendix B contains 7 tables and 1 diagram, so exploring the catalog is not easy as thought, because tables is not specialized obviously for each type, it is mixed and complex.

Now the engineer will read the catalog (Appendix B) to find which product suitable for his application.

The power supply which will apply all conditions and limitations mentioned before is RAW 1.5 KW. It takes a lot of time

to get this result, it is hard to read, relate, explore and find small information from this catalog. So another mechanism needed to save time and effort.

1.3 Computing Scientific applications

Scientific applications must be computed because of the difficulties in exploring and tracing systems through classical methods. Computing scientific applications will facilitate and quicken the following actions:

1- Exploring and tracing scientific application components.

2- Discovering relations between application parts and modules.

3- Maintenance of the application and determining the error location.

4- Developing the system.

Computing any application will stand basically on Database offered by the system. In fact scientific applications have the factors and elements which qualify it to be data based, then computed. These factors are:

1- Its characteristics and standards are data and can be stored.

2- Scientific applications partitioned into modules and structures, each module contain its own data. So in computer language it contains objects.

3- Structures of the scientific applications are related, so relationships exist in any application.

Data, objects and relations are the main components of any database, so scientific applications can be computed. Database theories will be used to serve in analyzing and drawing the data of any application

1.4 Scientific Systems Data Properties

Data in scientific applications will face some processes, these processes are:

1.4.1 Constraining: limitations and boundaries on data, these limitations can be divided into two parts:

A. Inherent constraints: means a certain data must be related in its value with another value. This means limitation comes from relation with another item.

For example figure 1-3 represents a block diagram for computer architecture. Now **The question is**: Can technician connect CD-ROM 16X with data bus width = 32-bit? The answer is

No, because data bus width of any device connected to PC like CD-ROM must be equal or greater than Local bus data width. The observer for block diagram will not easily notice that.

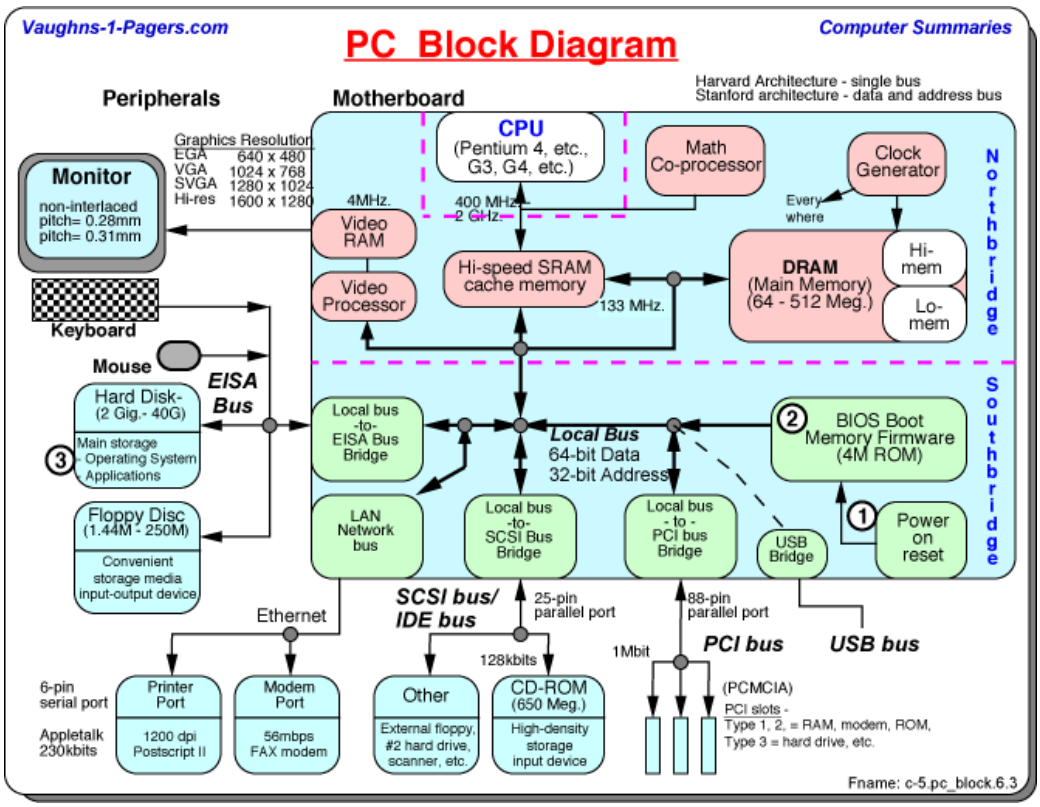
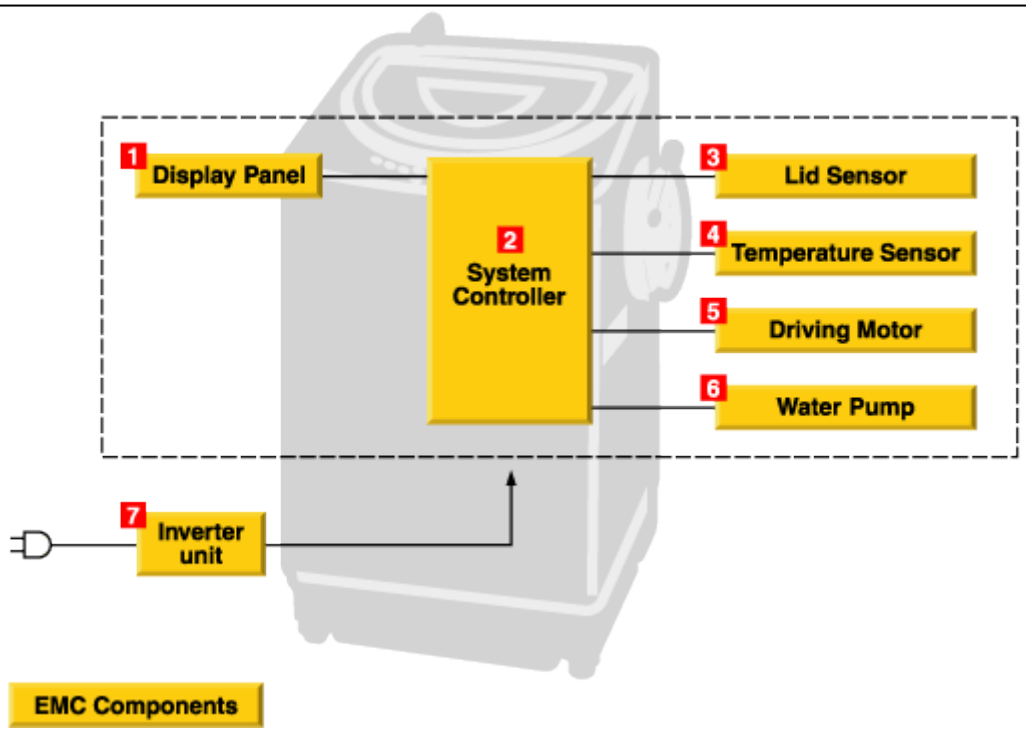


Fig 1-3 Pentium4 PC Block Diagram

So the data bus width for any peripherals connected to PC is limited by the data bus width of the main local bus. This limitation on peripheral data bus width is called inherent constraint in data.

B. Explicit constraints: restrictions and boundaries user puts on a certain item in data for scientific reason.

For example figure 1-4 shows a part of catalog supports washing machine structure, it contains block diagram and the components of inverter module.



Inverter Unit Components

Capacitor Type	Dimensions (mm)	Temperature characteristics	Rated voltage (V)	Capacitance range
CC45	5.5 to 16x5/6x5/7.5/10	SL	1 to 3	10 to 1,000pF
CD	7.5 to 15.5x8x10	E	250, 400	680 to 4,700pF
CS	12/16.5x7x7.5/10	E	250, 400	1,000 to 4,700pF

Inductor Type	Dimensions (mm)	Inductance range	Rated current
TSL0709	OD7.7x9.5	1 to 1,500microH	160 to 6,600mA
TSL0808	OD8.5x8.3	2.2 to 4,700microH	130 to 5,600mA
TSL1112	OD11.2x12.2	1.0 to 15,000microH	120 to 1,400mA
TSL1315	OD14x17	10 to 10,000microH	240 to 12,000mA

Fig 1-4 A part of Washing Machine Catalog

Note that the designer limited the following:

1- Rated Voltage for capacitors. Ex: CC45 voltage (1-3)v.

2- Capacitor Range. Ex: CD 680-4700 pF.

3- Inductance range for inductors. Ex: TSL0709 1-1500 microH.

4- Rated current for inductors. Ex: TSL1315 240-12,000 mA.

These borders for values were put by designer to make machine operating safely. It came from some mathematical calculations. It is an example for Explicit constraints.

1.4.2 Exploring Exploring means discover and survey the internal components and relations inside the system. User can explore through two ways which are:

a- Unlimited Exploring: it means to start reading the catalog and looking on block diagrams without specifications or concentrating on some elements. For example user wants to view all capacitors on inverter unit (return for washing machine catalog), so he will find:

Table 1-1 Unlimited Exploring

Capacitor Type
CC45
CD
CS

b- Limited Exploring: means that user wants to look on a specific thing with some conditions. For example (return to washing machine catalog) user want view only capacitors on inverter unit work on 400 v, so he will find:

Table 1-2 limited Exploring

Capacitor Type	Rated voltage (V)
CD	250, 400
CS	250, 400

1.5 Conclusion

Modern scientific systems are so complicated. Designers provide catalogs, maps and block diagrams to illustrate the system components and its work. Those illustration methods are hard to be understood and analyzed. Hence scientific applications must be computed to save time and effort.

Computing any system stands basically on database. Scientific application can be data based, and so computed. It has data, objects and relations, those are the factors for data basing.

Processes applied on applications data are exploring and constraining. So database and computer must serve in ease those processes.

Next chapter will discuss how to qualify catalogs and block diagrams to be data based, what is the requirements and the steps to setup database systems.

Chapter 2

Qualifying Scientific Data

2.1 Introduction

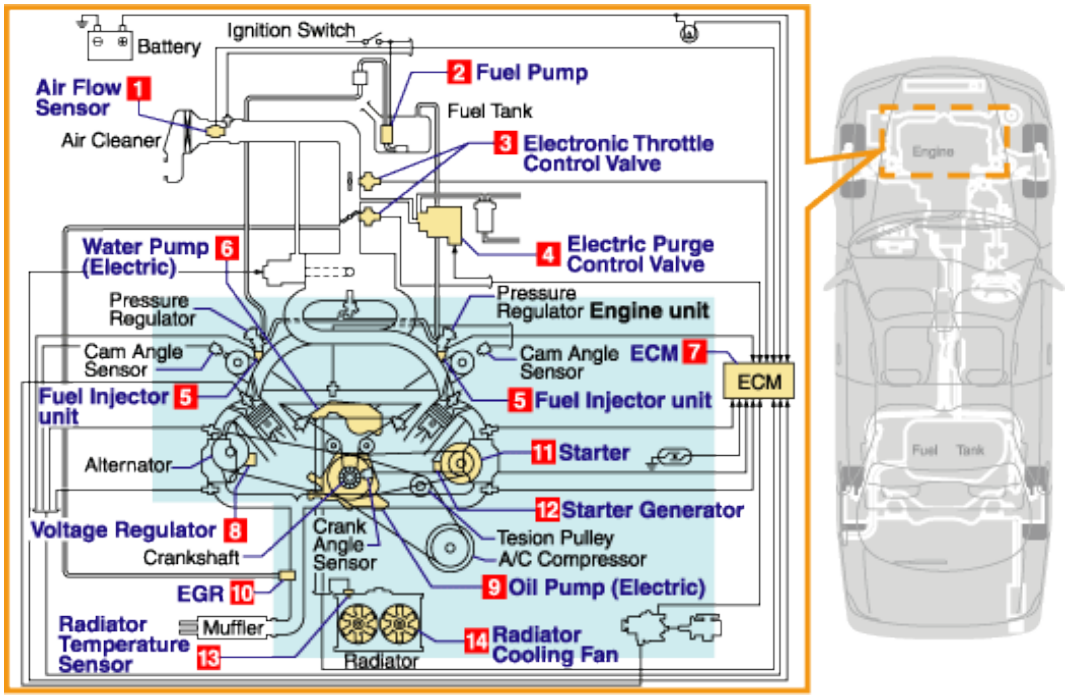
This chapter will discuss the steps needed to qualify scientific applications data to be a formal database capable for computing.

Usually engineers design catalogs and block diagrams to be understood from experts, but up to now data will be manipulated to give programmers the ability for analyzing, storing and programming.

This example related to engine control management (ECM) for a car. Figure 2-1 illustrates the block diagram for the ECM and the main elements inside each unit.

From database management viewpoint, If the data provided in tables have been stored as it, the following problems will be faced:

1. A lot of redundancy. For instance see chip capacitor how many times was it repeated.



(1)Air Flow Sensor	
	NTC thermistors NTCDS Series
	Chip capacitors C Series
	Chip beads MMZ Series

(2)Fuel Pump	
	Chip capacitors C Series
	Radial leaded capacitors FK Series
	Differential mode choke coils SF Series
	Inductors AML Series
	Ferrite magnets FB Series

(3)Electronic Throttle Control Valve	
	Geartooth Sensors GTS Series
	Chip capacitors C Series
	Chip capacitors for mid voltage C Series
	Rare earth magnets REC Series
	Ferrite magnets FB Series

(4)Electric Purge Control Valve	
	Geartooth Sensors GTS Series
	Sinter metal parts 3,M Series

(5)Fuel Injector unit	
	Chip varistors AVR-M Series
	Chip capacitors C Series
	Chip capacitors for mid voltage C Series
	Differential mode choke coils SF Series
	Common mode filters ACT Series
	Power inductors SLF Series

(6)Water Pump (Electric)	
	Chip capacitors C Series
	Differential mode choke coils SF Series
	Rare earth magnets NEOREC Series
	Ferrite magnets FB Series

(7)ECM	
	NTC thermistors NTCDS Series
	Chip varistors AVR-M Series
	Chip capacitors C Series
	Chip capacitors for mid voltage C Series
	Common mode filters ACT Series
	Common mode filters ZJYS Series
	Power inductors SLF Series
	Transformers SRW Series

(8)Voltage Regulator	
	Chip capacitors C Series

(9)Oil Pump(Electric)	
	Chip capacitors C Series
	Differential mode choke coils SF Series
	Rare earth magnets NEOREC Series
	Ferrite magnets FB Series

(10)EGR(Exhaust Gas Recirculation)

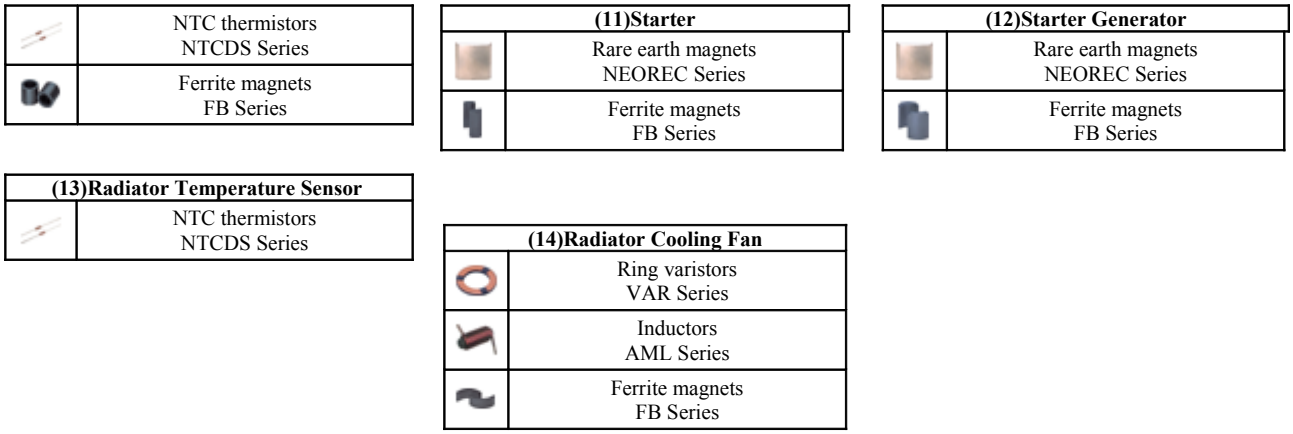


Fig. 2-1 Engine Control Management (ECM) catalog

2. No functional dependencies appear obviously for us. For instance does the unit “electric purge control valve” relate to the unit “ear flow sensor”?

3. Tracing the data is difficult.

4. Some important elements don’t appear clearly in tables. For instance ECM will not work without fuel or electricity. So an important functional dependency ambiguous in tables.

From this example and a lot of other examples the following absolutes for the modern scientific applications can be noticed:

A. One element may be shared through a lot of units.

(Redundancy)

B. Functional dependencies exist in every application.

C. Organic requirements needed for each application.

A new successful mechanism for drawing scientific data is suggested here, **it is suggested to partition scientific application data elements to the following objects:**

1. **Elements object: depend on their scientific unit (ex: Ohm, Farad, Centigrade)**
2. **Functional units object: includes the main units.**
3. **Organic requirements unit: like electricity, fuel.**
4. **Relations Object: defining relations between other objects**

This new mechanism will decrease redundancy, draws the relations obviously and can be computed easily.

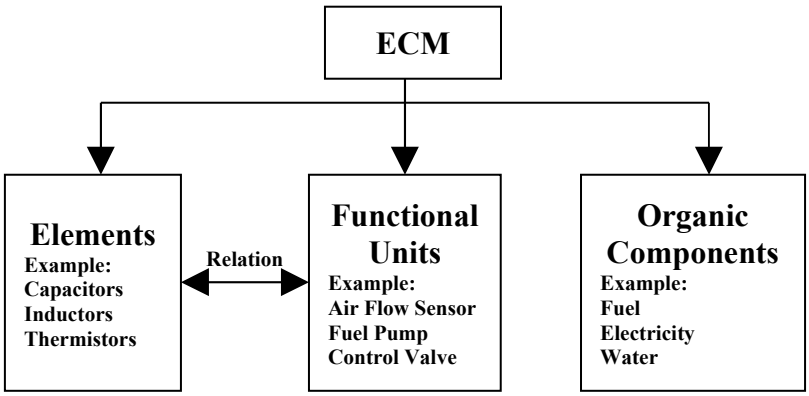


Fig 2-2 : New Suggested Generalization Mechanism

2.2 Elements Object: collect elements depend on their units. For instance F for capacitors, Ω for resistors and H for inductors. Hence collections will be as follows:

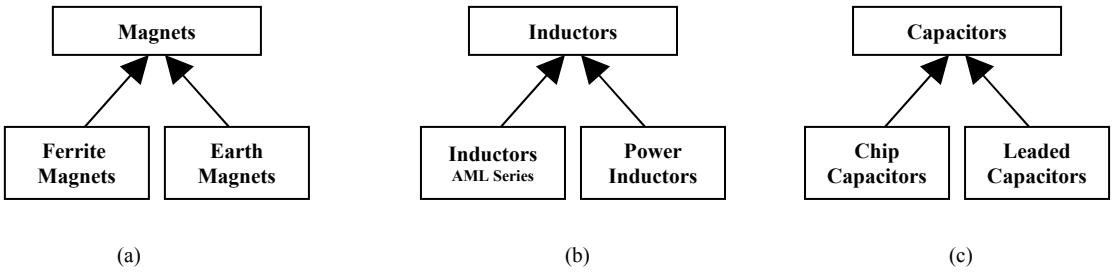


Fig. 2-3 Generalization depend on unit

In figure 2-3 it seems that ferrite magnets and earth magnets have the same unit which is Br (flux density unit), so it can be generalized in one entity "Magnets". And the same for AML inductors and power inductors have the unit H (Inductance unit) so it can be generalized in one entity (Inductors). Chip capacitors and leaded capacitors in figure c determined by V, ℃ and F which are (Voltage, Temperature and Capacitance units) so it can be generalized in one entity (Capacitors).

After generalizing the elements depend on their units we generalize again the new generated entities. See figure 2-4.

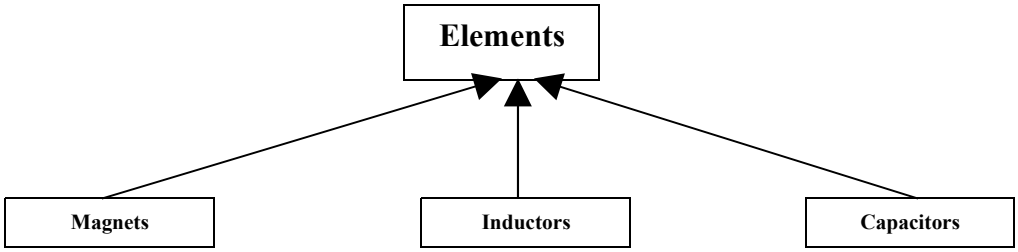


Fig. 2-4 Main and final generalization

2.3 Functional units object: generalizing depend on functions, so generalizing entities by abstracting air flow sensor, fuel pump, electronic throttle control voltage,….etc.

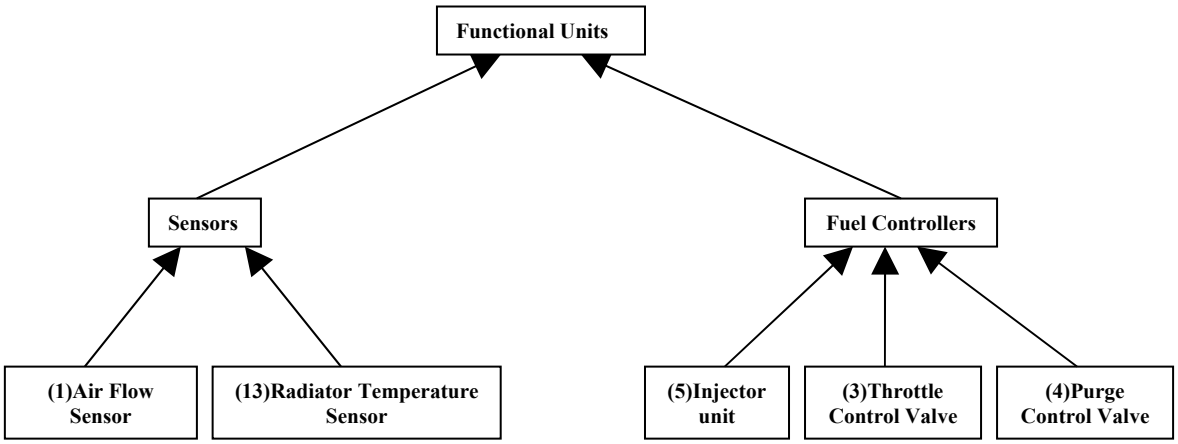


Fig. 2-5 Functional Generalization

2.4 Organic requirements unit: as a matter of fact any applications energy, it may be fuel, electricity, oil …etc, so those requirements cant be ignored in database model.

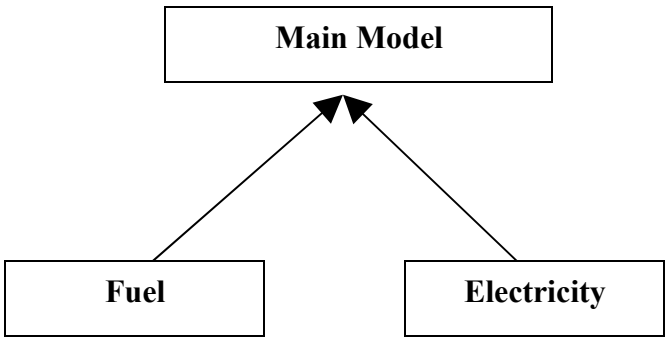


Fig 2-6 Organic requirements unit

Note that those requirements must be connected directly to the main grandfather of the model. Because there is no meaning to analyze model operation without those organics.

2.5 Relations Object: the relations between elements and functional units are very important to be registered. No way to repair or develop any machine without realizing the relation between its parts.

Returning to the previous example there is “radiator Temp sensor” and “radiator cooling fan”, here it is clear that a functional dependency exist, in fact it is a relation.

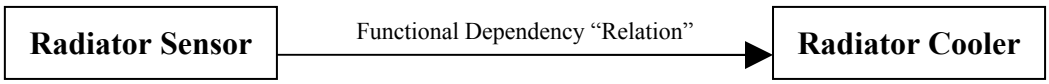


Fig 2-7 Functional dependency between units

Another example, the unit “EGR (Exhaust Gas Recirculation)” have many “elements”, hence a relation exist between “EGR Unit” and “Elements unit”.

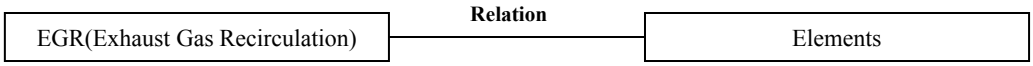


Fig 2-8 Relation between Units

We now generalize the units obtained in the last steps.

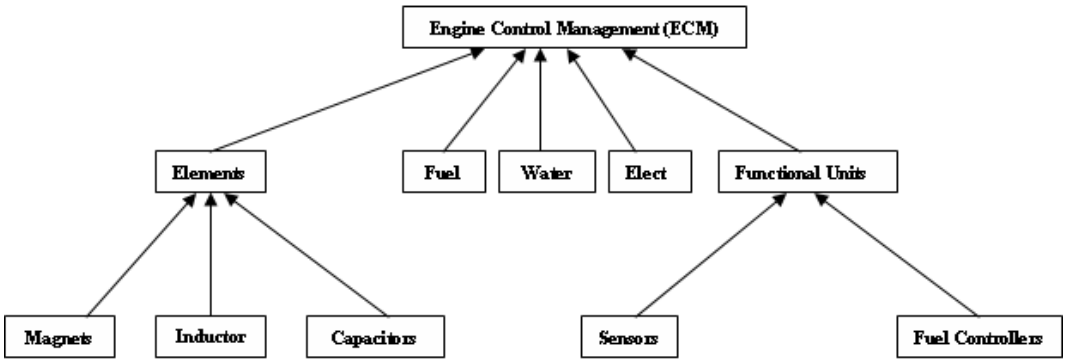


Fig 2-9 Full Generalized Data for a scientific application (Engine Control Management (ECM) for a car)

2.6 Scientific Data aggregation

It is the abstraction by which data is constructed form its constitute objects. For instance electronic elements can be characterized by its type, number and location on the machine. See figure 2-10.

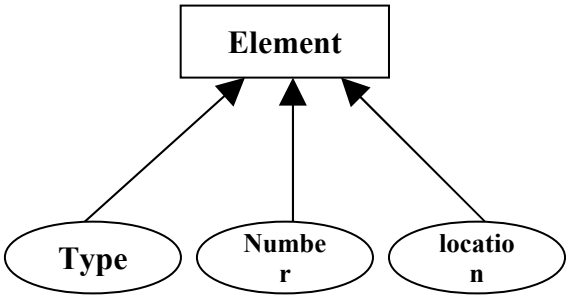


Fig 2-10 Data aggregation

Aggregation as a form of abstraction is very helpful because it gradually makes visible the structure of an object and how the

individual components of the object are related to it and to each other.

Example: a TV schematic describing TV components and unit functions. Schematic used for repair and explore the TV circuits.

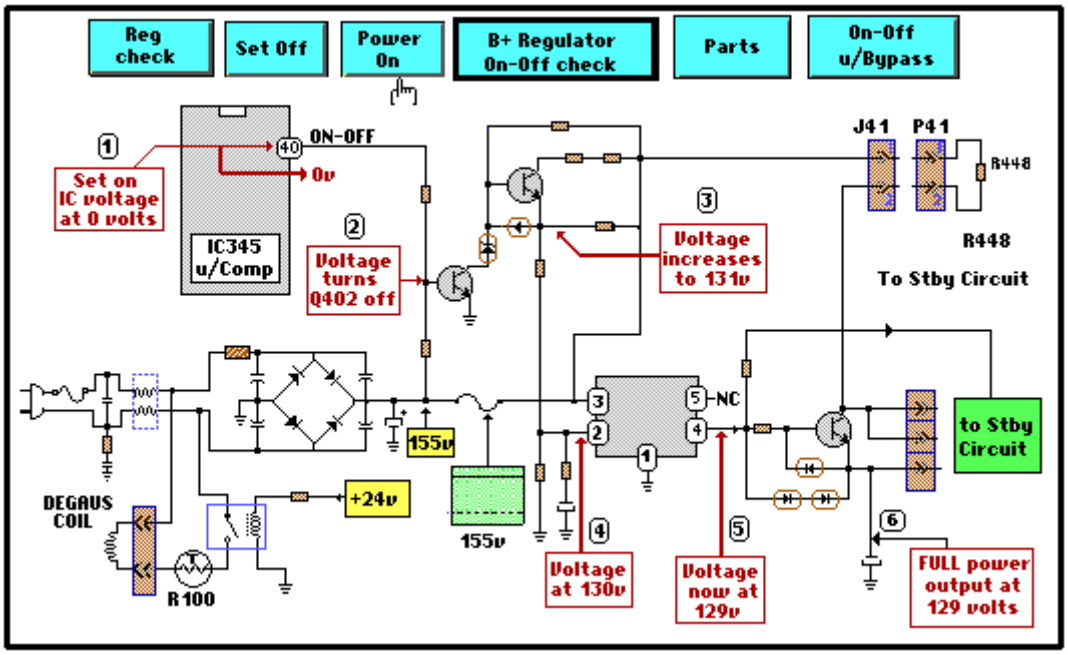


Fig 2-11 TV circuit diagram

Any one can see that it is hard to understand, analyze, or trace this map, so our strategy we will be applied to Redraw this map but from database viewpoint.

First: Elements Object

We gather the elements having the same property in one entity, so the list of entities in "Elements Object" are:

a- Resistors.

b- Diodes.

c- Capacitors.

d- Coils.

e- Transistors.

f- Fuses.

g- Crystals.

Second: Functional Units object

We now generalize circuits depend on functions, so the following Functional Units will be obtained:

1- Power Supply:

   a. Low Voltage Supplies unit.

   b. Standby unit.

2- Regulation Circuit Unit.

3- ON-OFF Unit:

   a. u/Processor Unit.

   b. Startup Unit.

4- Deflection Unit:

   a. Horizental Unit.

   b. Vertical Unit.

5- Audio Unit.

Third: Organic Requirments Object:

- AC power.

Hence our database view for our TV system will be:

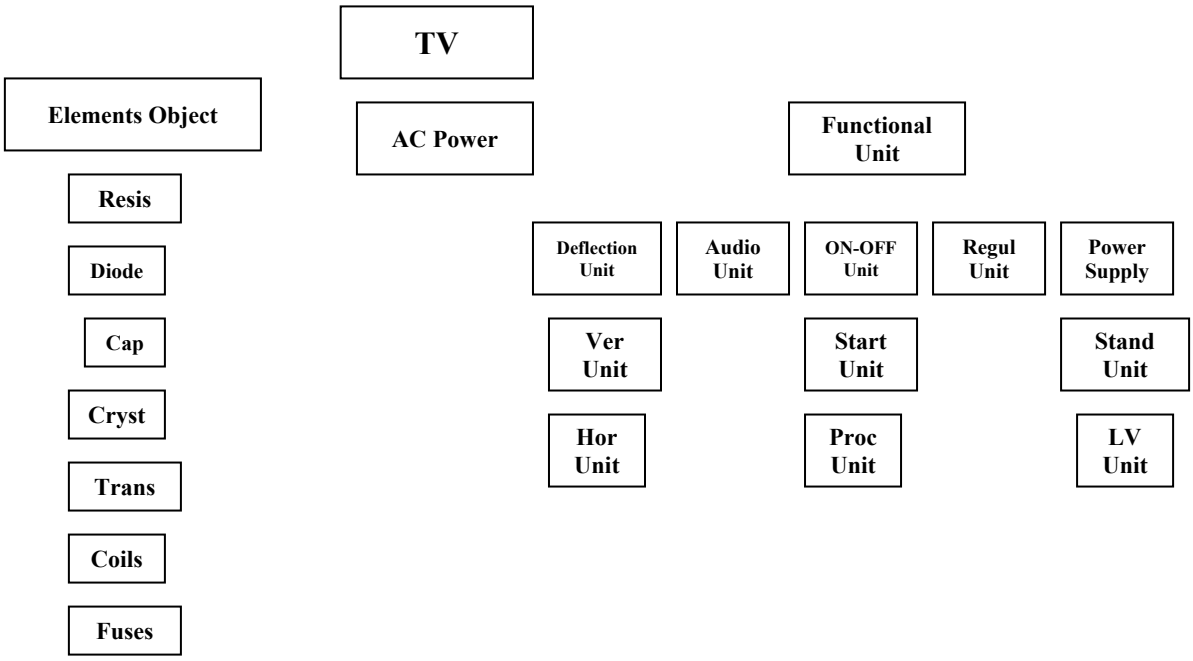


Fig 2-12 Database View for TV

Those above objects will be exploited by database management systems theories to be ready for computing.

2.7 Conclusion

In this chapter discusses how to rearrange scientific data in a manner suitable for database. A mechanism for analyzing scientific system will be produced; this mechanism stands on partitioning any scientific system to objects which are Elements, Functional and Organic units Object.

Chapter 3

Data Models Concepts

3.1 Introduction:

This chapter focuses on the definition of ***data***, ***database*** and ***data models***. Also this chapter outlines the factors taken in response when comparing between data models.

3.2 Main Concepts in Database

Data corresponds to discrete recorded facts about phenomenon from which we gain information about the world.

Data model: it is an intellectual tool that provides such an interpretation. Data model is a model about data by which a reasonable interpretation of data can be obtained. So it is an

abstraction device that allows us to see information content of data as apposed to the individual values of data.

Several data models used to represent data, but each data model has advantages and disadvantages depend on who is doing the schema design, and the field in which one is working [12].

Scientific system modeling aims to increase understanding some mechanisms in the process, and to optimize system behavior [15].

Schema: is a generic concept which identifies categories, their properties and relationships between them.

Database: a collection of data structured in particular way as related in a schema [17].

It is so important in data modeling to have the ability of hiding details and concentrate on general, that what is called The ***Abstraction*** of data base [14].

The abstraction of data comes through:

1- Obtain categories of data.

2- Combine categories into more general category.

Consistency can be reduced by eliminating redundant data. Redundancy Elimination leads to a simpler conceptual model which more accurately reflects the real world [13].

3.3 Data modeling levels

There are two levels in data modeling are:

1- Infological modeling level: based on creation a scientific model from user viewpoint. We get this level by what's called enterprise description.

Enterprise description describes the information requirements which used mainly for infological purposes.

2- Datalogical modeling level: focuses on design systems to achieve the information requirements from computer viewpoint. This level uses database description [17].

Data models created in the 1970^{s} increased the independency between data modeling and programming [6]. Hence one can design a data model for scientific applications without any background in programming.

The steps of designing a scientific database model are:

1- Abstract and understand concepts about application.

2- Capture all modules and parts contribute in the scientific system.

3- List the properties of these parts, is it integrated? or separated?.

4- Determine which model suitable for each system.

3.4 Structures:

Structures represent the parts of a system. they represent the main entities, their entries and the relationships between these entities [17].

Any data model will be represented by a graph or schema. Each model has its own elements which contribute to build the model.

Structures may be the same in all data models but it will take different names.

Not all structures are common in all models. There are some structures allowed and some of them disallowed in each model.

The main factor related to structure is ***structure complexity***. Lacking of complexity may be disadvantage because relationships not represented completely [13]. For example in Relational Model

the complexity is less than in Network Model, this means that less details in relational Model.

For example Network Model has a high structural complexity; this allows users to view a lot of relationships of a schema.

3.5 Constraints:

Constraints are the logical restrictions on data, its useful when it's generic, and not useful for a particular instance of objects.

The types of constraints used in data models:

1- Inherent Constraints: it specifies that all relationships in the hierarchal database are structured as trees [4]. for example the resistance current increases the temperature, the temperature determines the cooler speed, see Fig 3-1.

2- Explicit constraints: it provides a flexible mechanism for augmenting the structure specification of database [9].

For example user can define temperature must be less than 26°.

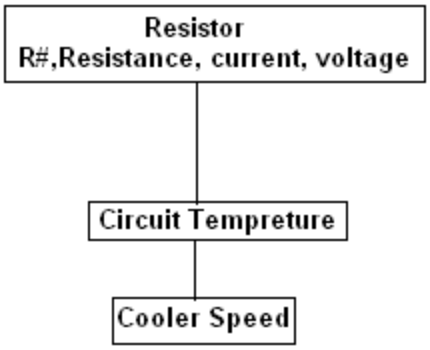


Fig 3-1: Inherent constraints

Constraint complexity is also very important. Constraint complexity can guide the users in interpreting the semantics of the database. For example the lack of constraints complexity in Relational Model allows meaningless relationships (joins) to be formed.

Other example: inherent constraints in Entity Relationship Model limit the data modeling capability and free unnatural organization of the data.

Explicit constraints appear desirable; they provide more flexibility in constraint specification than inherent constraints.

3.6 Operations:

Operations means processes applied by the user to get a specified data from the database, operations divided into two parts, ***Navigation operations and Specification Operations***.

Data models differs how it serves scientific operations. Some models provide ordering to its data, others don't. Selection data items will depend on the data model we use.

Navigation Operations: occurred when data language selects one object at a time by following a logical path through the database structure. Navigation Operation uses selection thru currency. For example, ask "get the next item in data" without specifying any property for this item.

Thru Currency: it’s a logical position in DB used as indicator to perform some actions.

The basic ways of setting currency in the database

1- Establish position in the database independent of the relationship between records.

2- Permit navigation among records, via connections, according to the links.

Specification Operations: selection of an item according to a specific property provided by the user [17]. For example "give me the temperature of the resistor where current = 0.2 mA", other example "give me the resistor where it's resistance more than 50KΩ.

Specification operation outputs all items - agreed with specified conditions- at the same time.

Chapter 4

Hierarchal Modeling for Scientific Applications

4.1 Introduction

At the moment we say hierarchal, we start thinking about past old and useless design. But when we say that IBM still depends

hierarchal in its information management applications we start rethinking and reviewing ourselves again.

Actually whether you want to build your own forum, publish the messages from a mailing list on your Website, there will be a moment that you'll want to store hierarchical data in a database. And, unless you're using a XML like database, tables aren't hierarchical; they're just a flat list. You'll have to find a way to translate the hierarchy in a flat file [1].

Storing trees is a common problem, with multiple solutions. There are two major approaches: the adjacency list model, and the modified preorder tree traversal algorithm.

This chapter will work on the second method which is “tree traversal algorithm”, because it is much faster and retrieving data in this method achieved with only one query. Finally almost everything can be done with this technique that could be done with the adjacency list method.

In this article, this method of saving hierarchical data will be explored. The tree for a TV schematic will be used as an example.

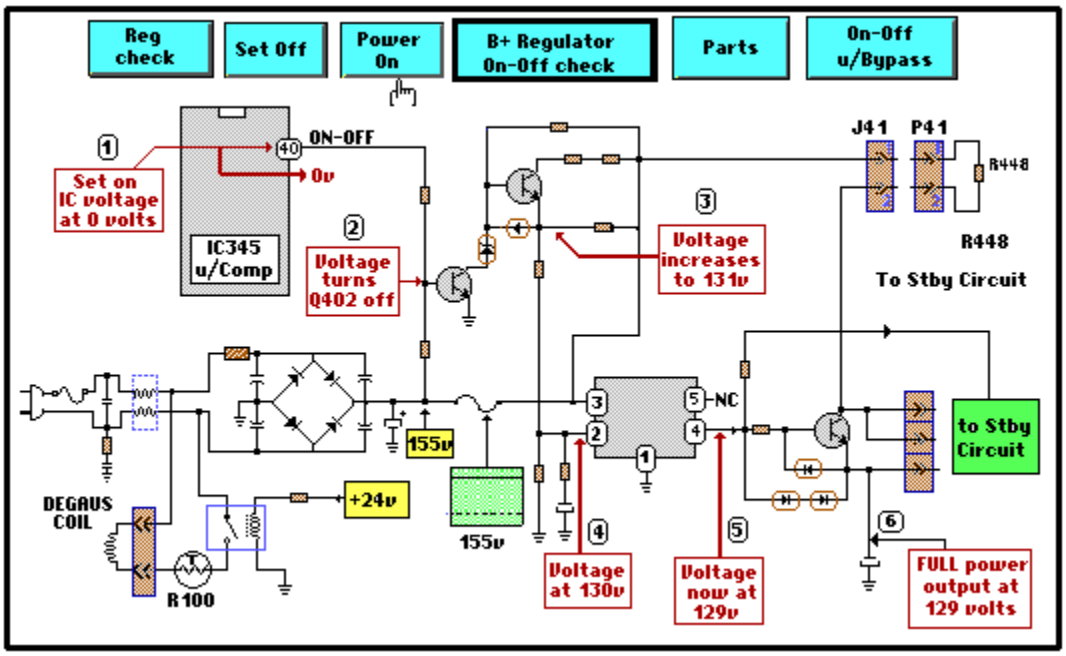


Fig 4-1: TV schematic

This schematic organizes its components by elements, by functional units and by organic components. The tree looks like this in figure 4-1.

This article contains a number of code examples that show how to save and retrieve data. This example is written in PHP. it can be probably easily to translate them to another language of choice.

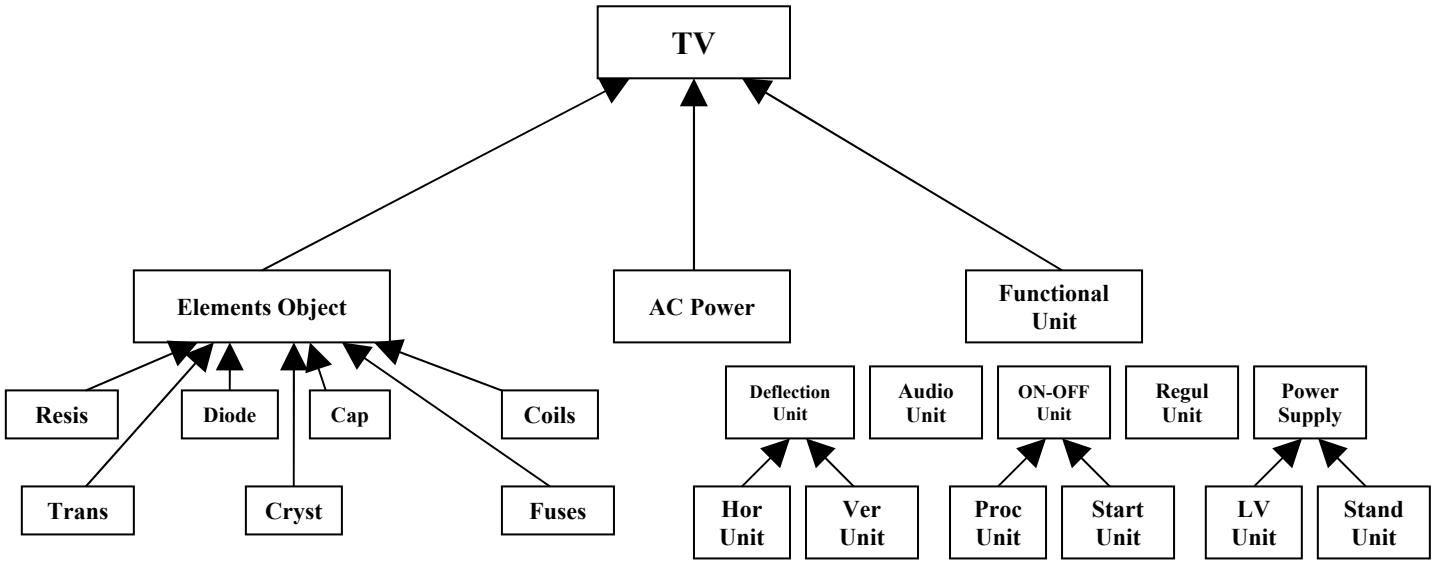


Fig 4-2 TV tree

Finally, Any scientific database model must respect two viewpoints which are:

1- Technicians: who will use this model in maintenance process. So they will concentrate on errors happen in the system and their reasons.

2- Developers: who will browse and explore this model to develop the application.

So the errors happen in any scientific application must be taken in response and ease the reaching for its main reason or reasons. So tree will be expanded to receive errors.

4.2 Modified Preorder Tree Traversal

Before starting scientific modeling, the expanded tree which assimilates problems and errors of our system will be shown.

The problems may happen in the TV system will be summarized and each problem will be given a code number:

Table 4-1 TV problems and reasons

TV Error	Code	Reason (Probabilities)
No Raster and No high Voltage.	001	a- Fuse
No Raster, high voltage cuts out, Relay clicks.	002	a- Start up. b- Regulator. c- Horizontal IC.
No Raster, we have high voltage, relay clicks.	003	a- ABL circuit b- Blanking circuit c- Picture tube d- Fuse
No Raster, no high voltage, relay clicks.	004	a- Horizontal IC.
No Raster, low voltage, relay clicks.	005	a- Regulator.
Full Raster, have picture, no sound, no Hum.	006	a- Regulator.
Full Raster, have picture, no sound, have hum.	007	a- Audio Unit.
Full Raster, have picture, sound cuts out erratic.	008	a- Circuit Board.
Bent vertical line picture.	009	a- Yoke. b- Vertical IC.
Vertical line.	010	a- Resistors. b- Diodes.
Vertical problem, pixels cut off .	011	a- Resistors. b- Capacitors.
Vertical Raster, no picture, no sound.	012	a- Tuner.
Vertical Raster, no picture, has sound.	013	a- Tuner.
Vertical Snow, no picture, no sound.	014	a- Tuner. b- Antenna.
Vertical Snow, no picture, no sound.	015	a- Tuner.

		b- Antenna.
Vertical present, no color.	016	a- Chrome IC.

We will try to put some of those errors at the end nodes in our tree, so each error connected with its related causing unit, so tree will be as follow:

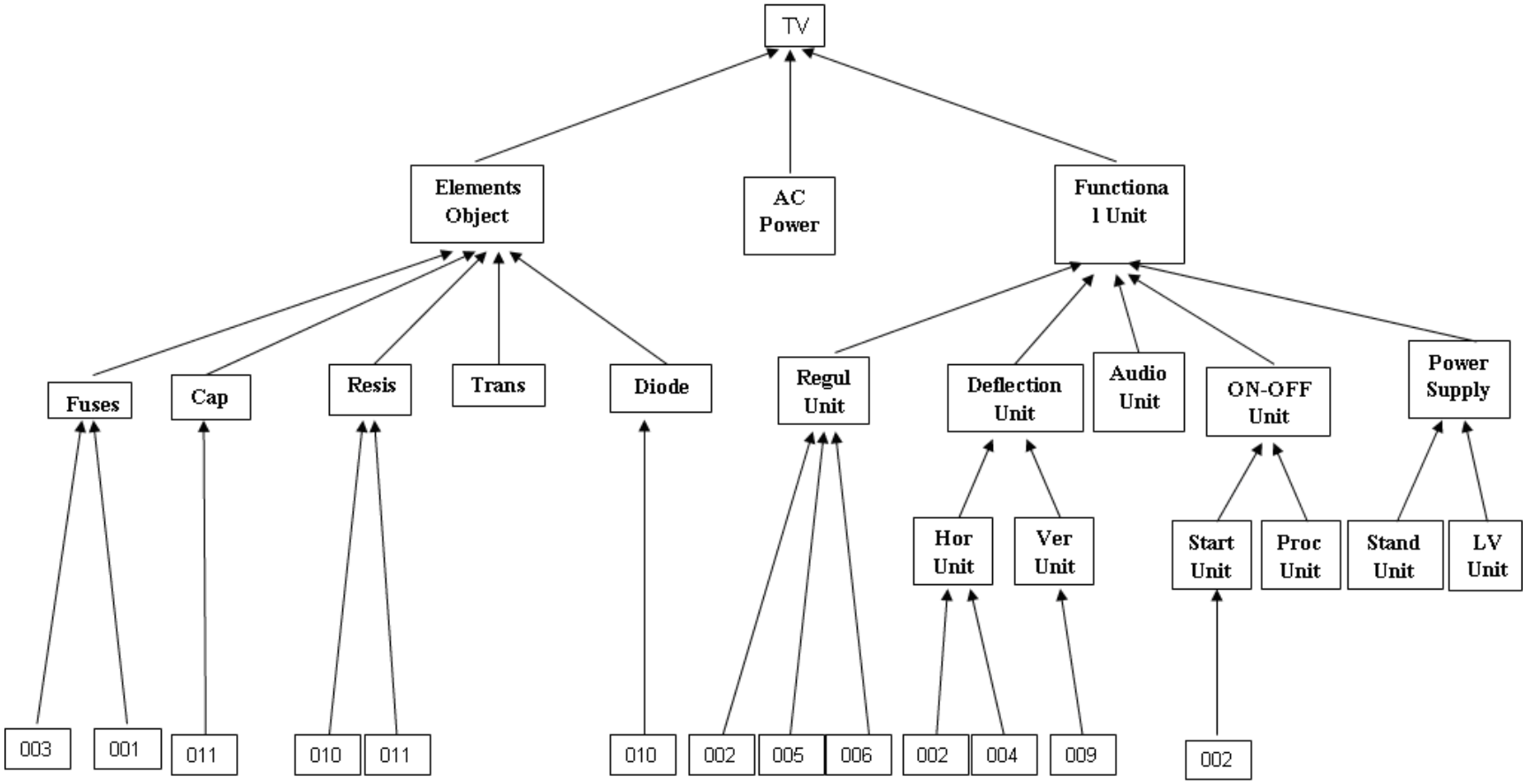


Fig 4-3 Part of TV hierarchal modeling

We'll start by laying out our tree in a horizontal way. Start at the root node (‘TV'), and write a 1 to its left. Follow the tree to ‘Elements Object' and write a 2 next to it. In this way, you walk (traverse) along the edges of the tree while writing a number on the left and right side of each node. The last number is written at the right side of the ‘TV' node. In this image, you can see the whole numbered tree. See fig 4-3.

Before continue, let's see how these values look in our table:

Table 4-2 Table expression for TV tree model

parent	title	Lft	rgt
	TV	1	66
TV	Elements	2	25
Elements	Fuses	3	8
Fuses	003	4	5
Fuses	001	6	7
Elements	Cap	9	12
Cap	011	10	11
Elements	Resis	13	18
Resis	010	14	15
Resis	011	16	17
Elements	Trans	19	20
Elements	Diode	21	24
Diode	010	22	23
TV	AC	26	27
TV	Functional	28	65
Functional	Regulation	29	36
Regulation	002	30	31
Regulation	005	32	33
Regulation	006	34	35
Functional	Deflection	37	48
Deflection	Horizontal	38	43
Horizontal	002	39	40
Horizontal	004	41	42
Deflection	Vertical	44	47
Vertical	009	45	46
Functional	Audio	49	50
Functional	ON-OFF	51	58
ON-OFF	Start	52	55

Start	002	53	54
ON-OFF	Proc	56	57
Functional	Power_Supply	59	64
Power_Supply	LV	60	61
Power_Supply	Standby	62	63

Note that the words ‘left' and ‘right' have a special meaning in SQL. Therefore, we'll have to use ‘lft' and ‘rgt' to identify the columns. Also note that the ‘parent' column not really be need anymore. The lft and rgt values used to store the tree structure.

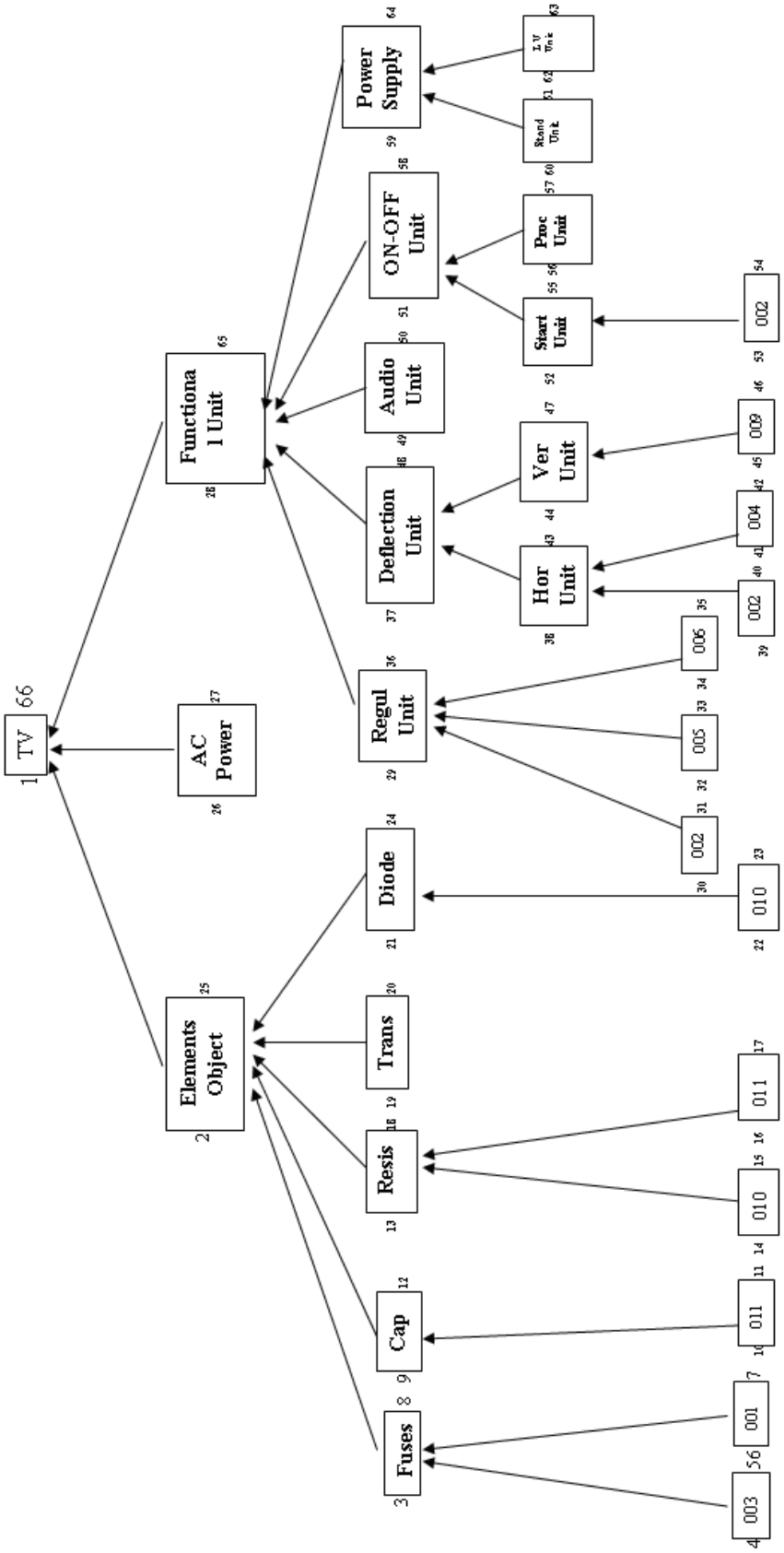


Fig 4-4 TV numbered tree "Tree Traversal Method"

4.3 System Error Detection

In the traditional methods like catalogues or schematics it takes a lot of time to relate between such an error and the circuit or unit responsible of this error.

With this new algorithm, we'll have to find a new way to get the path to a specific problem such as TV error easily and quickly. To get this path, we'll need a list of all ancestors of that error.

With our new table structure, that really isn't much work. When you look at, for example, ‘error 002' node which means "No Raster, high voltage cuts out, Relay clicks”, the question is which is the circuits may cause this error? The following quick and easy steps answer question:

1. know the left and right values of our error by the following statement:

Output for this expression

Table 4-3 output table in error finding

parent	title	Lft	rgt
Regulation	**002**	30	31
Horizontal	**002**	39	40
Start	**002**	53	54

you'll see that the left values of all ancestors are less than 30 or 39 or 53 while all right values are greater than 31 or 40 or 54.

2. To get all ancestors, this queries can do that:

```
SELECT FROM TREE  LFT, RGT
WHERE TITLE = “002”
SELECT TITLE FROM TREE WHERE LFT < 30 AND RGT > 31
ORDER BY LFT ASC;
SELECT TITLE FROM TREE WHERE LFT <39 AND RGT> 40
ORDER BY LFT ASC;
SELECT TITLE FROM TREE WHERE LFT <53 AND RGT> 54
ORDER BY LFT ASC;
```

Note that, just like in our previous query, An ORDER BY clause can be used to sort the nodes. This query will return:

Output:

Table 4-4 probabilities of error 002 reasons

TV
Functional Unit
Regulation Circuit
002

TV
Functional Unit
Deflection Unit
Horizontal Unit
002

TV
Functional Unit
ON-OFF Unit
StartUp Unit
002

We understand that error “002” may caused by the following circuits:

1- Regulation circuit which is a functional unit.

2- Horizontal circuit in the deflection unit.

3- Startup circuit in ON-OFF unit.

4.4 Adding New Scientific Unit to the System

Electrical designer discovered that a new problem may occur on Scientific system, he attends to define this new error to the system, can he do that? How a node can be added to the tree?

Or if designer wants to improve the system by adding a new unit, he needs to write a new manual and draw a new circuit diagram…etc. but up to now by using database design he needs only 2 – 3 SQL statements to improve and add the improvement to his electronic manual.

There are two approaches: you can keep the parent column in your table and just rerun the rebuild_tree() function -- a simple but not that elegant function; or you can update the left and right values of all nodes at the right side of the new node.

The first option is simple. You use the adjacency list method for updating, and the modified preorder tree traversal algorithm for retrieval. If you want to add a new node, you just add it to the table

and set the parent column. Then, you simply rerun the rebuild_tree() function. This is easy, but not very efficient with large trees.

The second way to add, and delete nodes is to update the left and right values of all nodes to the right of the new node. Let's have a look at an example. A new TV error which is “Audio cuts out” will be added, as the last node and a child of ‘Audio unit'. First, we'll have to make some space. The right value of ‘Audio Unit' should be changed from 50 to 52, the 51-52 ‘ON-OFF unit' node should be changed to 53-54 etc. Updating the ‘Audio Unit' node means that we'll have to add 2 to all left and right values greater than 49.

We'll use the query:

```
UPDATE TREE SET RGT=RGT+2 WHERE RGT>49;
UPDATE TREE SET LFT=LFT+2 WHERE LFT>49;
```

Now a new Error ‘Audio Cuts out' can be added to fill the new space. This node has left 50 and right 51. A title such as “040” can be given to the error.

```
INSERT INTO TREE SET LFT=50, RGT=51, TITLE=”040”;
```

Our new database will be as:

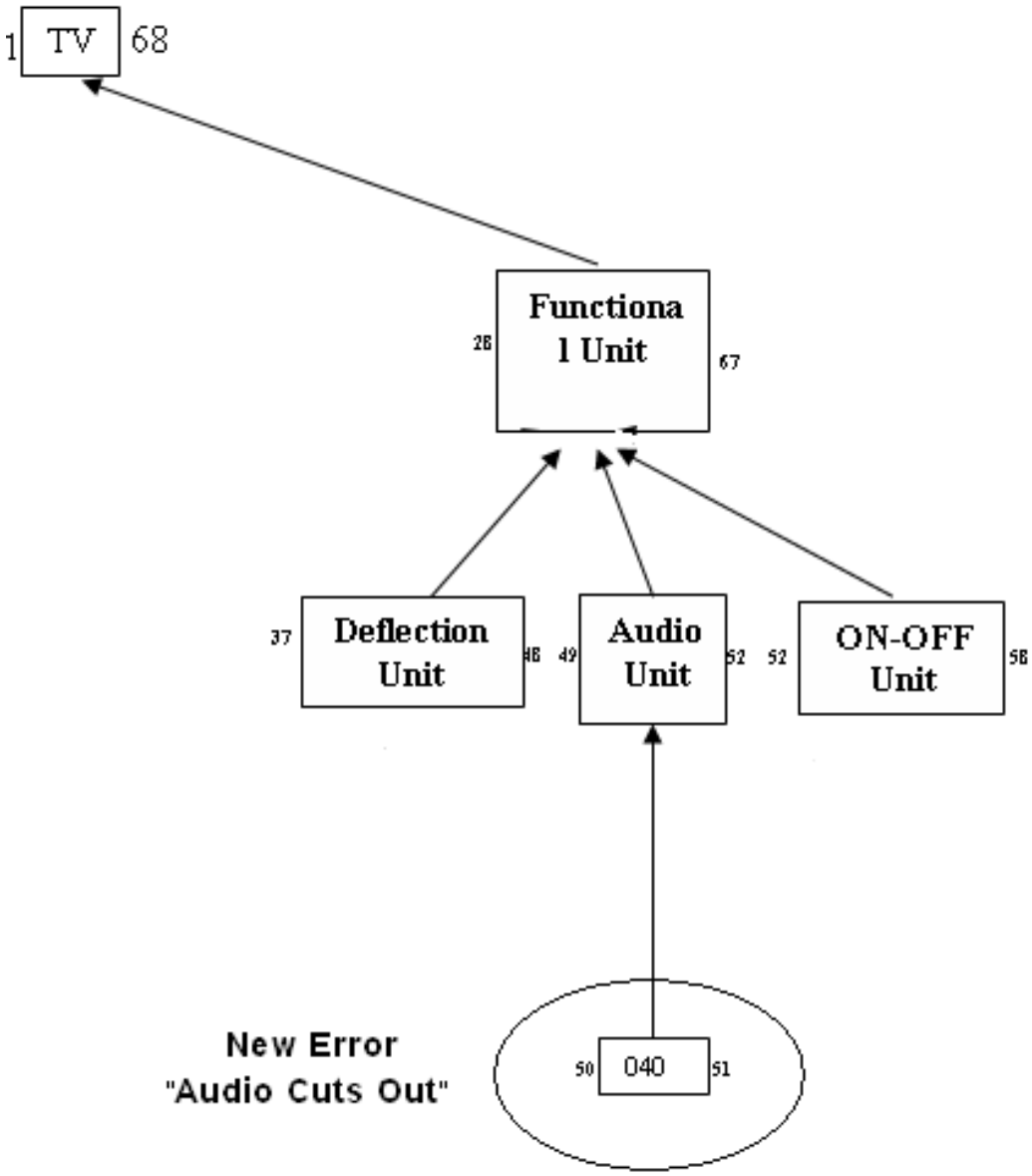


Fig 4-5 Adding new error “Audio Cuts Out” with code 040

4.5 How Many Descendants

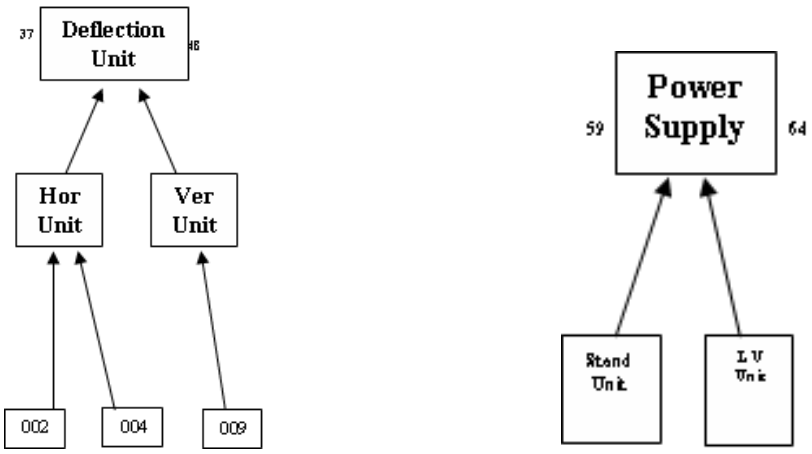
Can the designer know how many circuit or unit affected by a specific unit? Can he know how many errors happen if such unit or element damaged? This what we’ll discuss here.

If you give me the left and right values of a node, it is easily to know how many descendants it has by using a little math.

As each descendant increments the right value of the node with 2, the number of descendants can be calculated with:

DESCENDANTS = (RIGHT – LEFT - 1) / 2

With this simple formula, it is easily to know that “Deflection Unit” which has left value 37 and right value 48 has effecting 5 actions in our TV which are “Horizontal, Vertical, Error 002, Error 004, Error 009”. And that the “Power Supply” unit which has left value 59 and right value 64 affect two other circuits.



(a) (b)

Fig 4-6 (a) Deflection unit affect 5 units. (b)Power Supply affecting 2 circuits

4.6 Retrieve the Block Diagram Of a Scientific Application

Does hierarchal modeling serves technicians who wants to view the whole block diagram? Suppose that the developer

engineering wants to make improvements on the application, he must first view the application schematic, can he do that easily and quickly?

If you want to display the tree using a table with left and right values, you'll first have to identify the nodes that you want to retrieve. For example, if you want the ‘ON- Off' subtree, you'll have to select only the nodes with a left value between 51 and 58. In SQL, that would be:

SELECT * FROM tree WHERE lft BETWEEN 51 AND 58;

That returns:

Table 4-5 output of SQL commands

Functional	ON-OFF	51	58
ON-OFF	Start	52	55
Start	002	53	54
ON-OFF	Proc	56	57

Well, there it is: a whole tree in one query. To display this tree like we did our recursive function, we'll have to add an ORDER BY clause to this query. If you add and delete rows from your table, your table probably won't be in the right order. We should therefore order the rows by their left value.

SELECT * FROM tree WHERE lft BETWEEN 51 AND 58 ORDER BY lft ASC;

The only problem left is the indentation.

To show the tree structure, children should be indented slightly more than their parent. It can be done by keeping a stack of right values. Each time you start with the children of a node, you add the right value of that node to the stack. You know that all children of that node have a right value that is less than the right value of the parent, so by comparing the right value of the current node with the last right node in the stack, you can see if you're still displaying the children of that parent. When you're finished displaying a node, you remove its right value from the stack. If you count the elements in the stack, you'll get the level of the current node.

```
<?PHP
FUNCTION DISPLAY_TREE($ROOT) {
  // RETRIEVE THE LEFT AND RIGHT VALUE OF THE
$ROOT NODE
  $RESULT = MYSQL_QUERY('SELECT LFT, RGT FROM
TREE '.
                  'WHERE TITLE="'.$ROOT.'";');
  $ROW = MYSQL_FETCH_ARRAY($RESULT);
```

```
  // START WITH AN EMPTY $RIGHT STACK
  $RIGHT = ARRAY();

  // NOW, RETRIEVE ALL DESCENDANTS OF THE $ROOT
NODE
  $RESULT = MYSQL_QUERY('SELECT TITLE, LFT, RGT
FROM TREE '.
                'WHERE LFT BETWEEN '.$ROW['LFT'].' AND '.
                $ROW['RGT'].' ORDER BY LFT ASC;');

  // DISPLAY EACH ROW
  WHILE ($ROW = MYSQL_FETCH_ARRAY($RESULT)) {
    // ONLY CHECK STACK IF THERE IS ONE
    IF (COUNT($RIGHT)>0) {
      // CHECK IF WE SHOULD REMOVE A NODE FROM
THE STACK
      WHILE ($RIGHT[COUNT($RIGHT)-1]<$ROW['RGT']) {
        ARRAY_POP($RIGHT);
      }
    }
```

```
    // DISPLAY INDENTED NODE TITLE
    ECHO STR_REPEAT('  ',COUNT($RIGHT)).
$ROW['TITLE']."\N";

    // ADD THIS NODE TO THE STACK
    $RIGHT[] = $ROW['RGT'];
  }
}
?>
```

If you run this code, you'll get exactly the same tree as with the recursive function discussed above. Our new function will probably be faster: it isn't recursive and it only uses two queries.

4.7 Disadvantages Of Using Hierarchal Design In Scientific modeling.

Actually building hierarchal modeling for scientific systems will be hard in large systems, there will be a lot of nodes for the system tree, and a lot of values. But at the moment it has been built, maintenance and system tracing will be very easy.

Another disadvantage is that hierarchal modeling for scientific applications may be disable to relate parallel units with each other. For instance, it is difficult to relate Audio unit with deflection unit, those two units exists in different nodes.

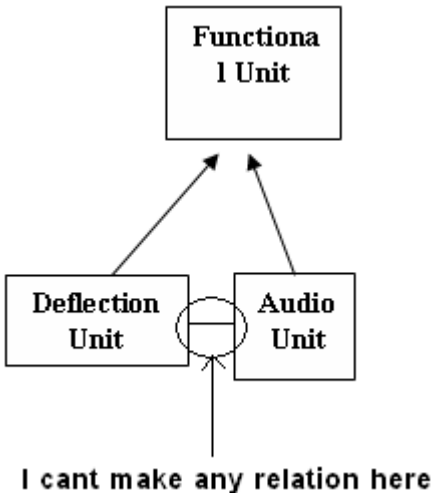


Fig 4-7 Relation between parallel units disallowed

Also in heirarchal modelling you cant relatem a specific error such as “002” error directly to more than one unit. This may lose more time in designing procedure.

4.8 Conclusion

Using hierarchal design saves a lot of time in error detection, scientific system improvements, viewing system structure. So it serves technicians in maintenance and engineers in design an develop.

Hierarchal design can be a good replacement for a hard copy of manuals, catalogues and circuit diagrams, which used to help in scientific systems analysis.

Some disadvantages in heirarchal modeling but it doest prhibite usin ehis kind of modeling, because it is not in basics of this modeling method. It is related only on time consumption.

Chapter 5

Using Relational Model in Scientific Applications

5.1 Introduction

In the Relational model, data is structured into simple tables. Relational model is the most common and distributed model.

Relational model is easy to design and there are a lot of programming languages manipulating it [4].

A relational database is more than just data organized into related tables. The relational database model is based firmly in the mathematical theory of relational algebra and calculus. The original concept for the model was proposed by ***Dr. E.F. Codd*** in a 1970 paper entitled ‘A Relational Model of Data for Large Shared Data Banks.’. Later Dr. Codd clarified his model by defining twelve rules (***Codd’s Rules***) that a database management system (DBMS) must meet inn order to be considered a relational database. In practice, many database products are considered 'relational' even if they do not strictly adhere to all 12 rules. Dr. Codd’s 12 rules is presented In appendix A.

Relational model considered a bottom–up approach because the steps of its creation are:

1- Select the attributes.

2- Combine these attributes into tables.

Relational model starts at elementary level of attribute [13].

A scientific example introduced here, this example will be manipulated through all models.

Scientific Example: suppose we have a robotic system. The system consists of power supply, infrared emitters, infrared detectors, and motors. The company which designed this system will record all data about the motors movements for this system [10].

There are three power cables exit from power supply; each of them provides different voltage and current. The three cables supplies electricity to all other devices in the system. The relations in this model as follow:

1. The motor speed depends on:
    - Power Cables which provide it with power.
    - Signals come from Infrared detector.
2. The infrared Emitter depends on Power Cable only.
3. The infrared Detector depends on Power cable and the Emitter.

So there are functional dependencies in this model. One motor can affect other motors speed. That what's called Recursive relationship.

Figure 5-1 shows the diagram of the circuit

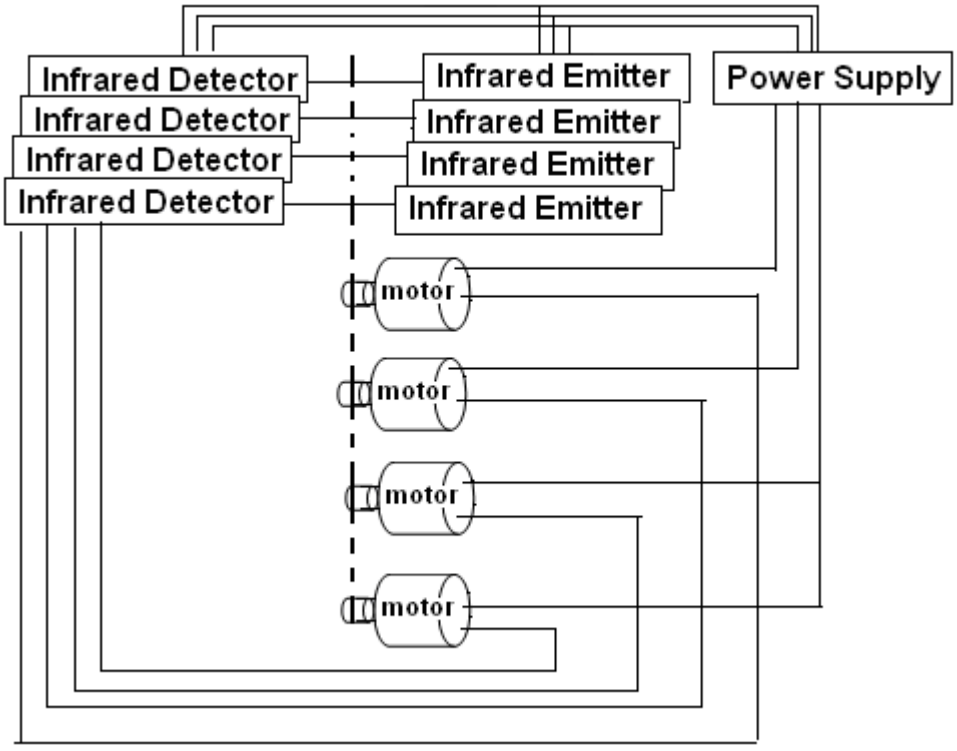


Fig 5-1: Diagram of speed control system in a robotic system

Table 5-1: Power Supply Entries

Power Cable	Voltage	Current	Location	# of Motors
10	5	1.2	1	2
20	10	1.5	4	1
30	15	2	7	1

Table 5-2: Infrared Emitter Entries

Infrared Emitter

Emitter Code	Location
15	2
16	3
17	5
18	6

Table 5-3: Infrared Detector Entries

Detector Code	Location
11	2
12	3
13	5
14	6

Table 5-4: Motors Entries

Motor Code	Location
21	11
22	8
23	10
24	9

5.2 Structures of Relational Model

The two basic elements in Relational model are ***Relational Schema*** *which* represents group of tables in the model, and ***Relation Scheme*** which represents one table.

Relationships between the tables can be represented by two ways:

1- Key propagation: adding attribute from one table to another. This done in 1:1 and 1:M relationships.

Example:

- Relationship between Power supply and infrared devices

  a- ***One*** power supply cable may provide ***many*** infrared emitters and detectors.

  b- ***One*** Infrared device provided by ***one*** power supply cable only.

This relationship called ***one to many*** relationship.

- Relationship between infrared emitter and detector

  a- ***One*** emitter relates to (emits to) ***one*** detector.

b- ***One*** detector relates to (receives from) ***one*** emitter.

This relationship called ***one to one*** relationship.

- Relationship between Power supply and Motors

  a- ***One*** power cable may provide ***many*** motors.

  b- ***One*** motor device provided by ***one*** power cable only.

This relationship called ***one to many*** relationship.

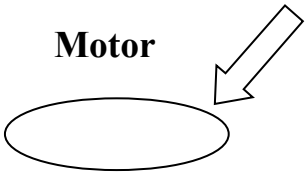
Table 5-5 shows the usage of key prorogation to represent the previous relationships. We added Power cable and Detector code to Infrared Emitter table to represent 1:M and 1:1 relationships

Table 5-5: Key Prorogation by adding Power cable and Detector code to Infrared Emitter table to represent 1:M and 1:1 relationships

Emitter Code	Power Cable	Detector Code	Location
15	20	11	2
16	30	13	3
17	10	12	5
18	10	14	6

Table 5-6 shows the representation of 1:M relationship between power cables and motors.

Table 5-6: one to many representation by Key progagation



Motor Code	Power Cable	Location
21	20	11
22	30	8
23	10	10
24	10	9

2- Separate relation scheme: new table consist of related relationship. This done in M:M relationships.

Example: Relationship between infrared Devices and motors

a- ***One*** infrared detector may control ***many*** motors.

b- ***One*** motor can be controlled by ***many*** infrared detectors.

This relationship called ***many to many*** relationship

Table 5-7 shows the representation of M:M relationship between Detectors and Motors by a new separated table.

Table 5-7: The representation of M:M relationship between Detectors and Motors.

Motor Code	Detector Code
21	11
22	11
23	12
24	13
21	12
24	12
22	14

Figure 5-2 shows the Relational schema for the scientific example

Figure 5-2: Relational Schema for Scientific Application

Power Supply

Power CableVoltageCurrentLocation# of Motors1051.21220101.5413015271

Infrared Emitter

Emitter CodePower CableDetector CodeLocation1520112163013317101251810146

Motor

Motor CodePower CableLocation2120112230823101024109

Figure 5-2: Relational Schema for Scientific Application (cont'd)

Infrared Detector

Detector CodePower

CableLocation11202123031310514306

Detector Motor

Motor CodeDetector

Code211122112312241321122412221 4

Suppose an error occurred and motor 21 stopped, by using the model it's easy to search for the power cable which supplies this motor, and the detectors which control this motor.

The set of the tables is called Relational Schema, one of the tables (Power Supply, or Infrared Detector, or…..) is called relation scheme.

5.3 Constraints in Relational Model

The constraints in Relational Model are:

1. No duplication rows are permitted [9].

This obtained by what's called ***Primary Key***, which is any subset of attributes having a value uniquely identify the row in a relation. For example primary key in Power Supply table is Power Cable, in Infrared Detector it is Detector Code.

2. The ordering of rows and columns is not significant in Relational Model.

3. No inherit constraints; This presents difficulties in terms of exploiting meaningful relationships between relations.

For example if user deleted Cable 10 from Power Supply table, other tables will not affected. It seems meaningless that Motor 23 and 24 supplied by Cable 10, so it is disadvantage of Relational model.

4. Provides a facility for specifying explicit constraints on the relations, these explicit constraints are:

a- Scope: limiting the domain of the value. For example specify that the attribute contents are integer, real, character.

b- Assertion: Predicates that specific condition must be satisfied by the user to the database. It can describe the permissible states for the data.

For example in power supply table, the voltage must be greater than 5v and less than 50v, 50>Voltage>5,

Another example in power supply table, the summation of #of motors must equal the number of motors in Motor table which equals four. Figure 5-3 explains this example:

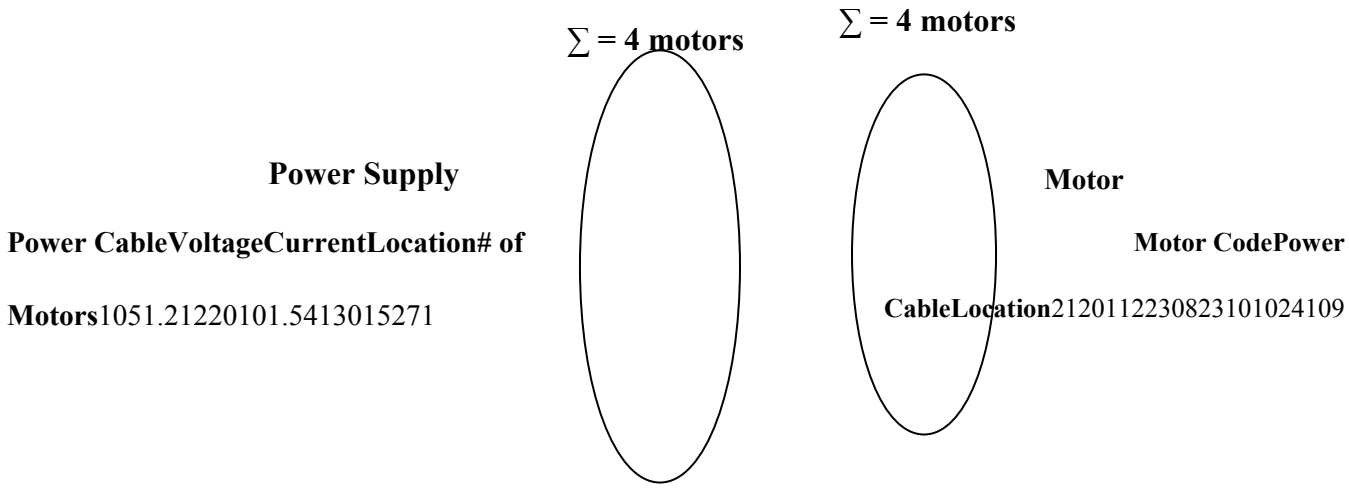
Example: Assertion by SQL Language to limit voltage in Power Supply table.

ASSERT C1 **ON** POWER SUPPLY:

Voltage >5 **AND**

Voltage<50

Fig 5-3: Assertion constraint.



5.4 **Operations on Relational Model**

Two basic operations which are Navigation operations and Specification operations applied to this model will be discussed separately.

5.4.1 **Navigation Operations on Relational Model**

Navigation means browsing data one by one without a specific condition. In order to navigate data in the Relational Database, two basic concepts needed:

a) Currency on the table.

b) An order of retrieval Data.

Example: Suppose an employee wants to navigate all motors in Motor table and there locations. The navigation process achieved by the following SQL commands.

1. CREATE SCAN Motors Code ON Motor
2. SET SCAN Motors Code
3. GET NEXT Motor Code
4. Output SELECT FROM Motor Code: Motor Code, Location

The steps of this program are:

1. line1 states that:

a. The name of this scan (navigation) is ***Motor Code***. The scanning name assigned to be referred at any position in the program.

b. Determine the table which will be browsed (Motor table).

2. Line2 indicates to start navigation; this will reset the currency indicator to the top of the table.

3. Line3 controlled the position of the currency indicator to navigate the following Motor.

Actually repeating of this process is needed to scan all Motors and this achieved by simple *for loop*.

4. Line4 asked to retrieve the ***Motor Codes*** founded through the scanning.

The output of this operation is concatenated in the same order indicated by the list of attribute names in the table. Table 5-8 shows the output (when using *for loop* for the scanning process):

Table 5-8: The output of navigation operation

Motor Code	Location
21	**11**
22	**8**
23	**10**
24	**9**

We achieved currency in line 1,2 and 3. This happened by determining the table (line1) to be navigated, then the position of the currency indicator in this table (line2) to start navigation from, after that controlling the position of currency indicator to move down to the following row in the table (line3).

Ordering of the output retrieved from the program depends on the ordering of rows in the original table (line4).

5.4.2 **Specification Operations on Relational Model**

The main feature of languages serves Relational model is its ability to define a new relation based on existing relations, using relational algebra or similar types of operations.

For example someone wants to know which Power Cables from the Power Supply table where power (Watts) provided (Current X Voltage) is over 15 W, then the user will write the following SQL lines:

- SELECT Power Cable FROM Power Supply

- WHERE Current * Voltage ≥ 15

This selection method is specified to a condition, so it can be said that it's a specification operation. The output of these commands will appear as a table, this table considered as a picture only, so it is called snapshot relation. Snapshot will stay only during the time of running the program; the output relation appears as follow:

Table 5-9: output of a specification operation

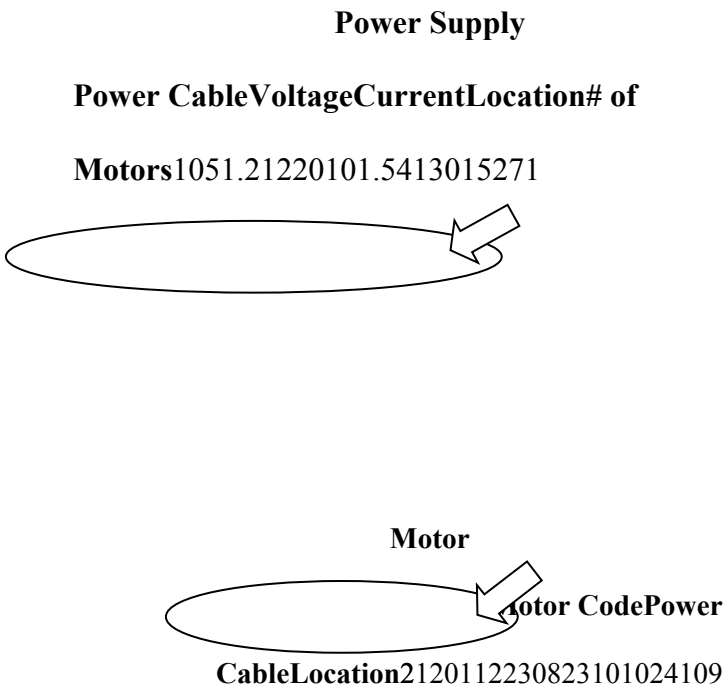
Power Cable
20
30

Example: The engineer in the company asking for the Motor Code which receives Current over 1.8A, the commands in SQL:

- SELECT Motor Code FROM Motors
- WHER Power Cable = (SELECT Power Cable FROM Power Supply WHERE Current > 1.8A)

Figure 5-5 explains the selection statements in the example

Fig 5-5: Compound selection



The output table will be:

Motor Code

22

Fig 5-6: output of the example

Quires can be manipulated by short statements, it is better than catching millimeter or looking to a map for a long time to determine which power circuit provides this device, or which devices related to this cable.

5.5 **Conclusion**

Relational Data Model depends on creating tables only, no relationships explained between these tables. Relational Model is easy and works well in relatively simple scientific situations [13].

Relational Model achieved a great goal; they have brought together the practitioners who implement the scientific systems and the researchers who conceptualize about them [17].

The most obvious concepts missing from Relational Data Models deal with the specification and representations of constraints on and among relations [5].

Chapter 6

Using Entity Relationship Model in Scientific Applications

6.1 Introduction

Entity Relationship model (ER) is a type of data model based on tables and graphs, it facilitates database design by allowing Enterprise schema [17].

Enterprise Schema represents the entire enterprise view of data and the documentation of the logical properties of these data. It is independent of storage or efficiency consideration.

ER is considered a conceptual model via top-down approach which starts from identifying entities and relationship types, and then uses these to construct a framework into which the attribute may be sorted [13]. The steps of creation Scientific ER model are:

1- Select the main entities in the scientific application.

2- Determine the relationships between these entities.

3- Assigning the entities with its related attributes.

6.2 Structures of Entity Relationship Model

The basic elements of the ER model is

1- Entity Set: which represents the tables, it is represented by rectangular. Figure 6-1 shows entity sets

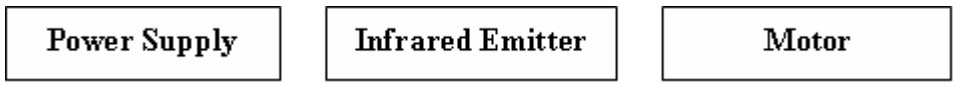


Fig 6-1: Three entity sets

2- Relationship Set: which represents the relationship between entities, it is represented by diamond.

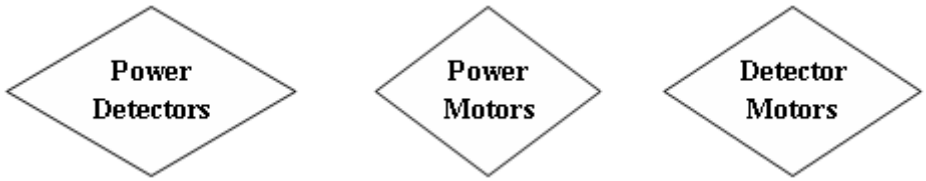


Figure 6-2: Three Relationship sets

3- Label on the connecting arc (1, M): it represents the maximum cardinality permitted for an entity set in a relationship set. For example label 1 between Power Supply and Power Motor sets means that at maximum one power cable can provide the same motor. And also label M between

Motor and Detector Motors means that many motors can be affected by the same detector.

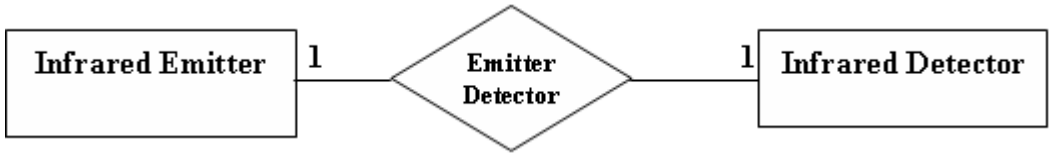


Fig 6-3: labels on Relationships, one to one relationship

ER model depicted by a diagrammatic technique called Entity Relationship Diagram (ERD), figure 6-4 shows the ERD for our scientific example in chapter2:

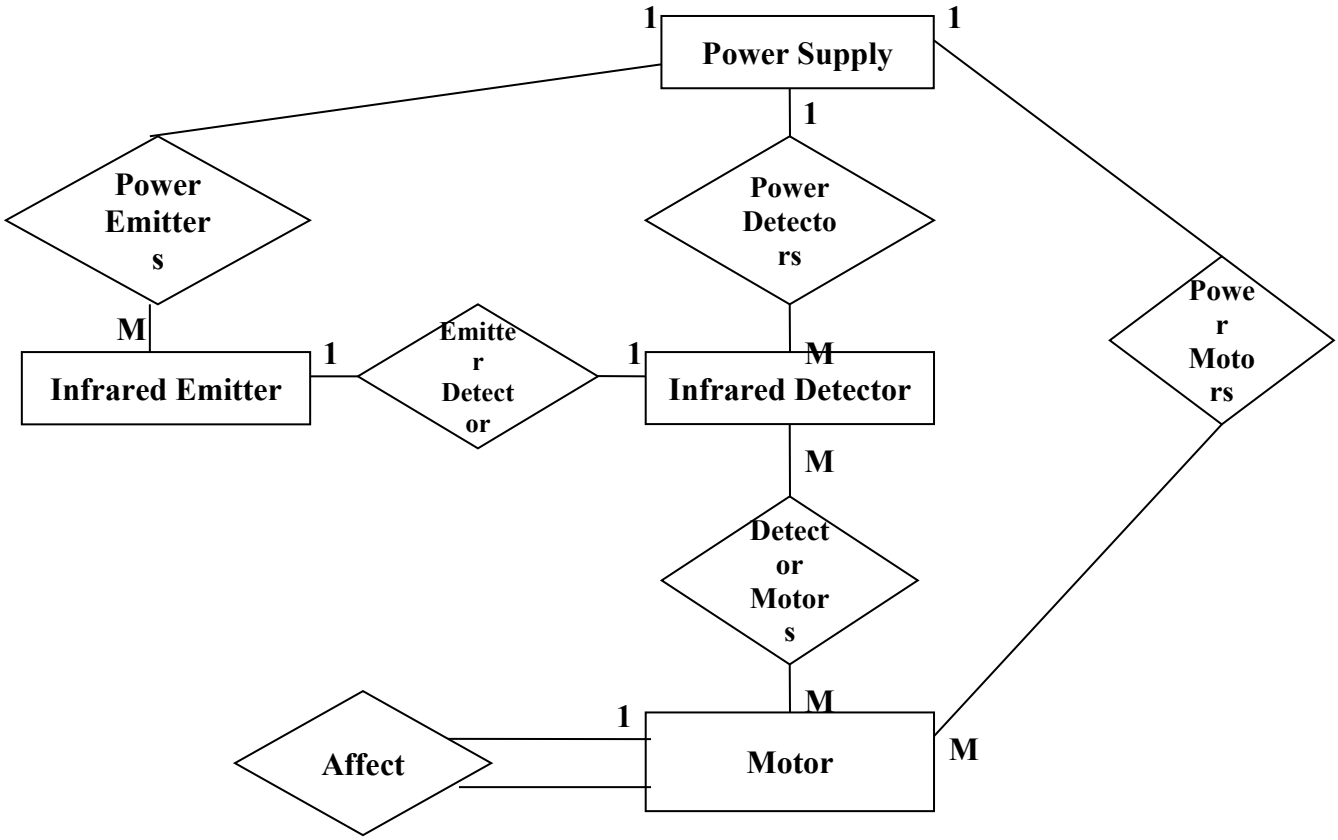


Fig 6-4: Entity Relationship Diagram (ERD)

The main Properties of ER Data Model structures:

1- Links can represent 1:1, 1:M, M:M relationships between entities.

2- Recursive links are allowed. For example the Motor speed affects the speed of other many motors.

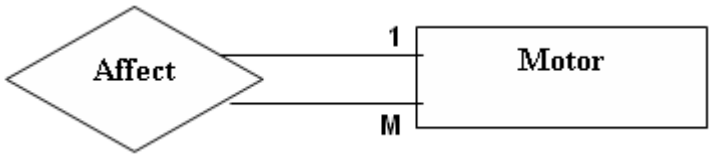


Fig 6-5: Recursive Link

3- It is possible to have more than one relationship set between the same two entity sets; figure 6-6 shows an example.

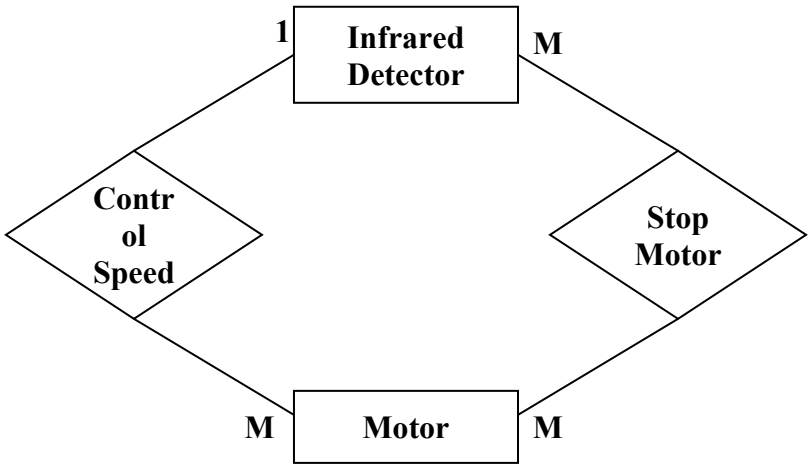


Fig 6-6: Two relationships between the same entities

This property gives more details about Relationships. In the figure 6-6 the first relationship 1:M means that a motor speed controlled by one detector only. The second relationship M:M means that a motor stopped by many detectors.

4- A relationship set may be among n entity sets; figure 6-7 shows an example.

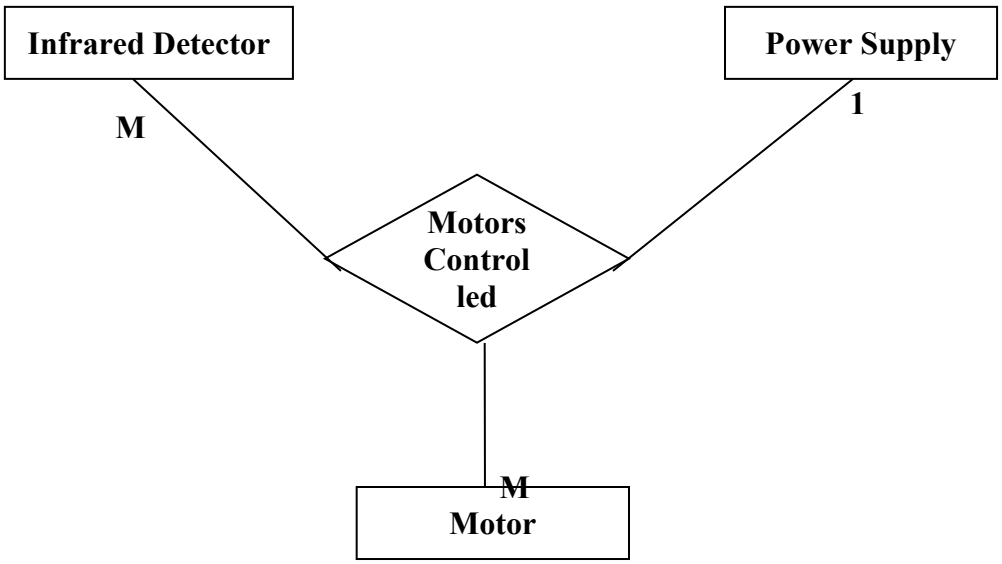


Fig 6-7: Three entity sets (Ternary Relationship)

The figures explains three entities related by the same relationship which is controlling the motor work. The relation in figure 6-7 indicates that:

1. One detector supplied by one power cable affects ***many motors***.
2. One motor supplied by one power cable controlled by ***many detectors***.
3. One motor affected by many detectors supplied by **one power cable.**

Creation of an ER model in a programming language means the following must be created:

1. Creation of main entity sets and its attributes. In the example it was:
    a. Power Supply: (Power Cable, Voltage, Current, Location).
    b. Infrared Detector: (Detector Code, Location).
    c. Infrared Emitter: (Emitter Code, Location).
    d. Motor: (Motor Code, Location).
2. Creation of the Relationship sets and assigning attributes for each set. In the previous scientific example the relationship sets and their attributes will be as follow:
    a- Power Detectors: contains Power Cable, Detector Code.

b- Power Emitter: contains Power Cable, Emitter Code.

c- Emitter Detector: contains Emitter Code, Detector Code.

d- Power Motors: contains Power Cable, Motor Code.

In scientific models no way to separate between the elements forming the model. One of the scientific model properties is the integrity. It is meaningless that each part at the same model works alone without feedback, this property makes the decision of ER model is the favorite [15].

6.3 Constraints in Entity Relationship Model

Constraints offered by ER model are explicit constraints. The user put them in the definition of the database:

1. Value set: it means the domain and the range for each attribute, for example number, character, real, date from 1 to 1000.

2. Existence dependency: It means that the existence of the member entity in a relationship set depends on the existence of the owner entity for the same relationship set.

It is very important restriction for scientific applications because there are many functional dependencies in scientific applications.

Example: The motor depends on the power cable. When the power cable removed the motor removed automatically from the database. This concept served by the languages compatible with ER model [16].

Figure 6-8 shows this type of constraints, it is represented by labeled box, double-rectangle, and label E in the relational set.

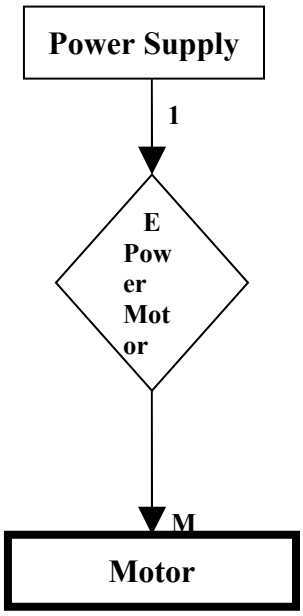


Figure 6-8: Existence Constraint

3. ID Dependency: it occurs when an entity cannot be identified alone, it can be identified only by its dependency with other entities.

Example: Infrared Emitter identifies Infrared Detector. User can't identify any infrared detector without identifying emitter; figure 6-9 shows this type of constraints.

ID Dependency is an Existence constraint but the existence constraint is not an ID Dependency.

As example the ID Dependency in the scientific example is that when deleting Infrared Emitter then Infrared Detector which relates to it removed automatically.

ID dependency represented by labeled box, doubled rectangle and ID label in the relationship set, Fig 6-8 shows this type.

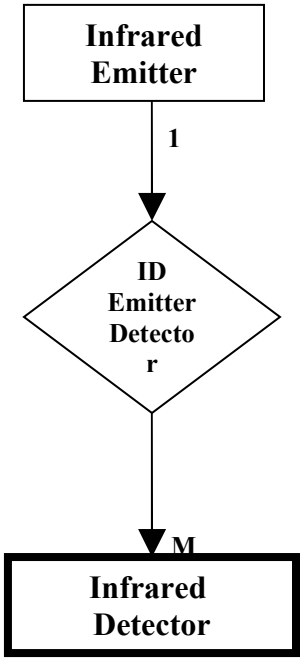


Fig 6-9: ID Dependency Constraint

We conclude that the constraints in the entity relationship support the integrity property of any scientific model. It is better to use ER model in complex models because it prohibits any meaningless operations [8].

6.4 Operations on ER Model

ER model offers a specification operations through specificified data languages such as CABLE (Chain Based Language). Languages based on the fact of links and paths exist on

ER model. The complexity of a retrieval specification is greatly reduced.

Example1: some one wants to browse all motor Codes exist in the database, so by CABLE language:

- Select Motor.Motor Code.

Example2: Power cable 30 will be separated for some time, so the engineer must know which motors affected? as a matter of quick answer the electrical engineer will refer to motors table and say that cable 30 provides motor 22 then motor 22 only will be affected; but in fact power cable 22 affects infrared detector 12 and 14 and these detectors affect the motors 21, 22 and 23.

In this scientific example it is clear that there are two paths between power supply and motor, the first path is direct connection, and the second path is through Infrared Detector.

Figure 6-10 explains the two paths between power supply and motor. Network language follows power cable 30 in these two paths. It is amazing, easy and quick to know this information by the following few CABLE statements

1) OUTPUT Motor.Motor Code

2) Select Power.Power Cable = '30'

3) Select Power.Power Cable = '30' /Infrared Detector

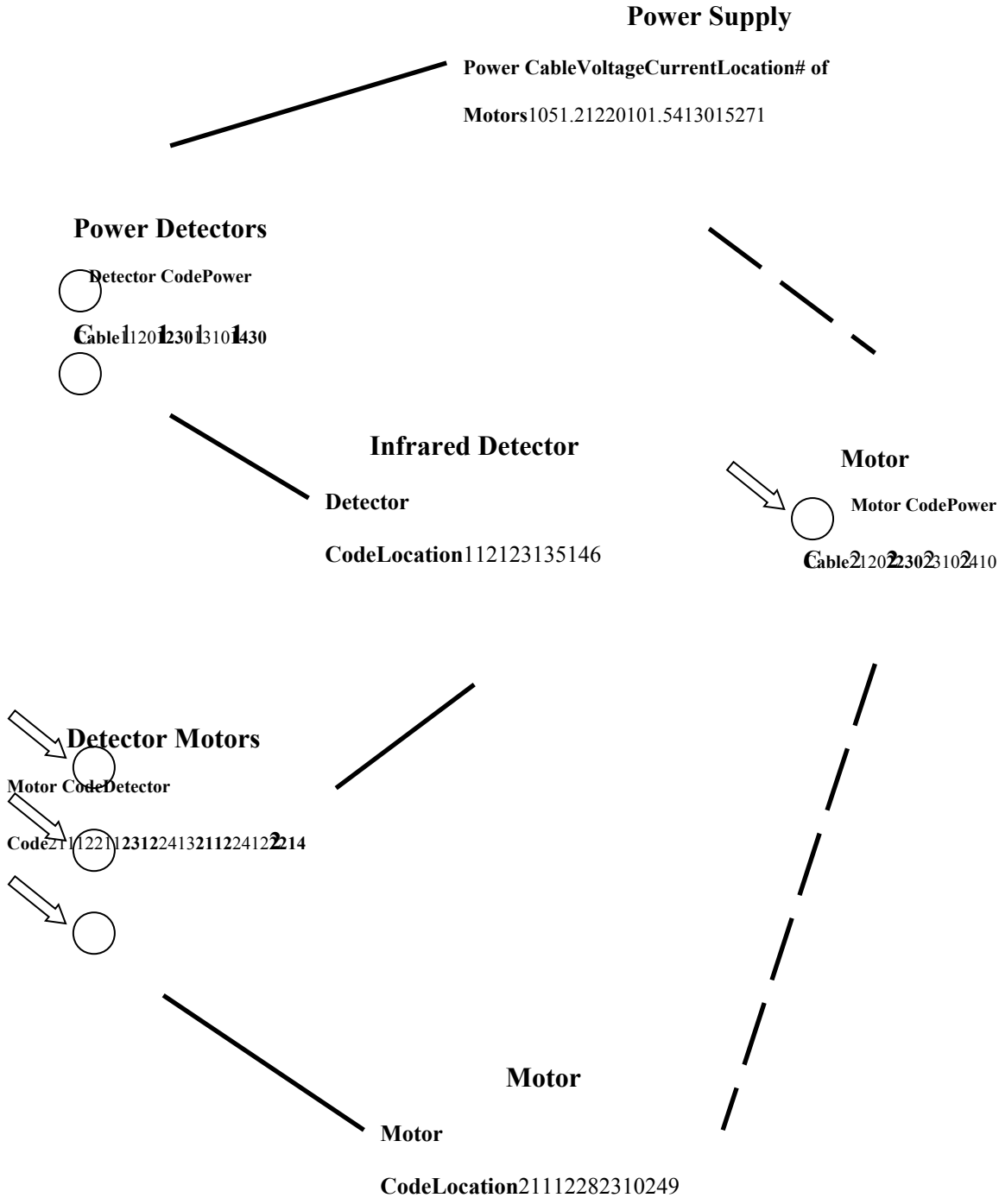
Line 1 states: show and display all motor codes retrieved from the following paths.

Line 2 states: through the direct path between Power Supply and motor tables, select all rows that have power code=30. The output of this command line is (this output will not appear to the user):

Motor Code	Power Cable
22	30

Fig 6-11: Output of the line 2 command, selection in the direct path

Fig 6-10: Two paths between Power Supply and Motor



Line 3 states: pass through Infrared Detector table path and select all rows in this path having power cable = 30, the output of this command line is (this output will not appear to the user):

Detector Code	**Power Cable**
12	30
14	30

Motor Code	***Detector Code***
21	12
22	14
23	**12**

Fig 6-12: output of the line 3 command, selection through indirect path

The Motor Codes appear in the two selection statements are Motor Code 21, 22 and 23, these codes are the output of this program.

6.5 Conclusion:

Entity Relationship (ER) model satisfies the requirements of scientific database design method; it makes integration between the entities and the relationships between entities. ER model is general enough and semantically rich enough to express the structures and constraints of the scientific applications.

ER model can be designed for computer implementation or for human understanding to a scientific model.

ER constraints make a powerful and flexible scientific representation system, it restricts any error may occur by the user.

Operations applied to ER model are specification operations. It is easy and efficient to be applied on any scientific system. Specification operations retrieves data wanted in high accuracy by simple statements.

ER Model uses the Mathematical Relation Construct to Express the Relationships between Entities. The relational model and the ER model both use the mathematical structure called Cartesian product. In some way, both models look the same – both use the mathematical structure that utilizes the Cartesian product of something. A relationship in the ER model is defined as an ordered tuple of "entities." In the relational model, a Cartesian product of data "domains" is a "relation," while in the ER model a Cartesian product of "entities" is a "relationships." In other words, in the relational model the mathematical relation construct is used to express the "structure of data values," while in the ER model the same construct is used to express the "structure of entities."

ER Model Contains More Semantic Information than the Relational Model. By the original definition of relation by Codd, any

table is a relation. There is very little in the semantics of what a relation is or should be. The ER model adds the semantics of data to a data structure. Several years later, Codd developed a data model called RM/T, which incorporated some of the concepts of the ER model [7].

Chapter 7

Using Network Data Model in Scientific Applications

7.1 Introduction

Network data models based on tables and graphs, it is not far from ER model, some different will appear between ER model and network model.

Network model graph seems as tree. It seems to be complex, and it disallows some relationships which is important for scientific applications.

7.2 Structures of Network Model

The basic items forms network models are the record type which represents the table and the set type which represents the relationship between entities.

The diagram represents network model called Data Structure Diagram, it contains record types and set types.

In network model data structure diagram hasn't tables, this differs from relational model where its representation must include the tables.

Figure 7-1 shows a scientific network model represented by Data structure diagram.

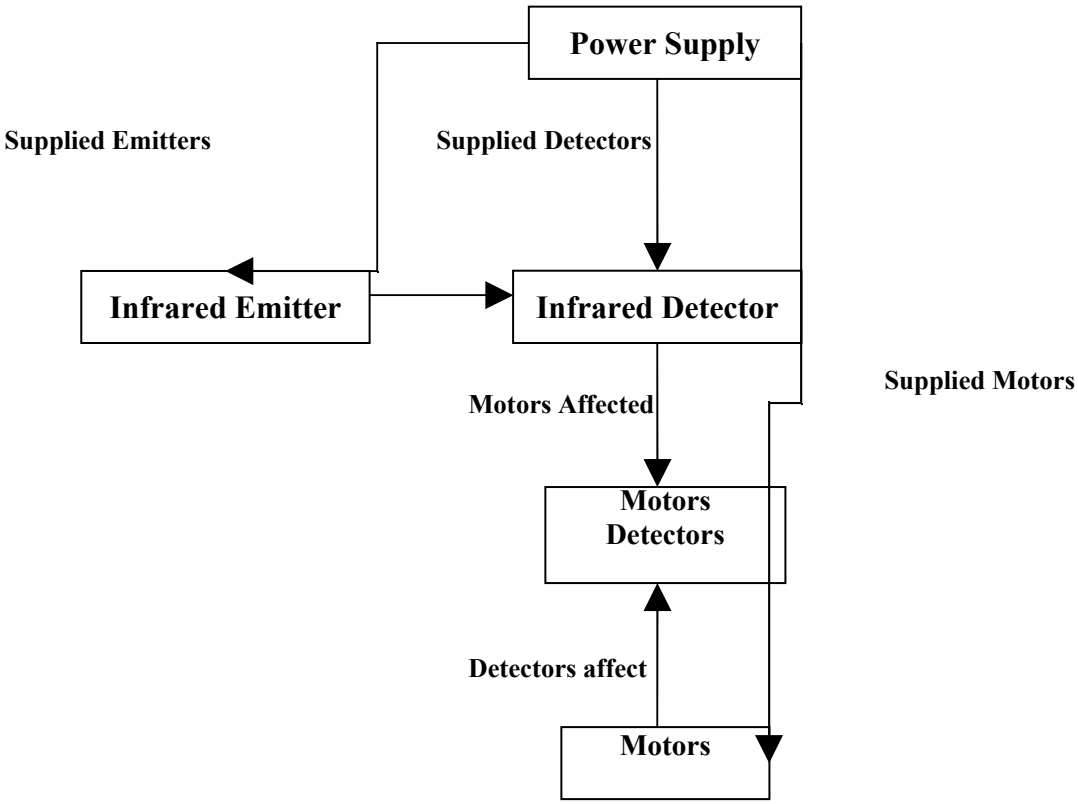


Fig 7-1: Data Structure Diagram for a Network Scientific Model

The record types appear in this model are Power Supply, Infrared Detector, Infrared Emitter and Motors.

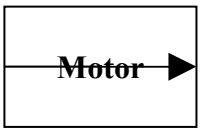
The Set types in this model are supplied detectors, Supplied Emitters, Supplied Motors, motors affected and Detector affects.

Properties of the network model structures are:

1. Only functional links (1:1, 1:M) between entities allowed in network model. M:M Relations cannot exist between two related records directly, this will be discussed in the following section.

2. For any set type their must be OWNER and MEMBER. Owner always at the tail of the arc, while member at the head of the arc.

3. Recursive Link disallowed in network model, it impossible here to say that motor affect other motors, it is one of the disadvantages of this model. Figure 7-2 shows recursive link.



Affect

Fig 7-2: Recursive Link disallowed in network model

When defining the structures of a scientific Network model, it is recommended to define first Record types and attributes included in each type. Then defining the set types and specifying the OWNER and MEMBER for each set as the following commands (uses ANSI Data Definition Language):

RECORD NAME IS Power Supply

Power Cable

Voltage

Current

Location

RECORD NAME IS Infrared Detector

Detector Code

Location

RECORD NAME IS Infrared Emitter

Emitter Code

Location

RECORD NAME IS Motor

Motor Code

Location

SET NAME IS Supplied Motors

OWNER IS Power Supply

MEMBER IS Motor

SET NAME IS Supplied Detectors

OWNER IS Power Supply

MEMBER IS Infrared Detector

SET NAME IS Supplied Emitters

OWNER IS Power Supply

MEMBER IS Infrared Emitters

Actually the nature of entities (OWNER or MEMBER) is very important for scientific applications. This property gives limitations on the operations applied to the database.

Definition of OWNER or MEMBER as a structure increases the integration in the scientific model.

For example it is disallowed to delete the OWNER while there is a MEMBER related to it. User can't delete Power Cable 10 while Motor 23 and 24 still related to this cable.

In the Relational model, there was no OWNER or MEMBER, so any changes to any table will not affect other tables.

In ER models also OWNER and MEMBER missed. Instead, in ER model user determines the type of dependency if it is ID or Existence dependency.

7.3 Constraints in Network Models

Constraints offered by this model are:

1. The first constraint is inherit constrain. It is a Built-In constraint can be applied to any scientific database by the user. It states that any member record can have at most one owner record.

Example: a computer company attending to make a central computer. This computer has multiprocessors, multi hard disks, multi CD-ROMs and multi coolers. As a matter of fact, computer coolers work always depending on power supply working. Coolers don’t depend on CPU or hard disk. When using the last constrain

user can't make a relationship between motherboard and cooler. Figure 7-3 explains this constraint.

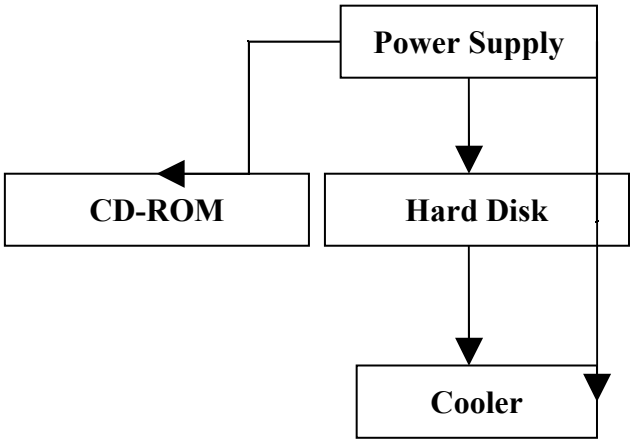


Fig 7-3: Cooler is member for two owners (Power Supply, Hard disk), this linking can be restricted by programmer.

Any user tries to relate (make a table) hard disks and coolers will face restriction message from the program.

2. Many to Many relationships cannot be represented directly; it can be represented by indirect way. Figure 7-4 explains this property.

The disability of representing M:M relationship is disadvantage.

In scientific applications there are a lot of many to many relationships. But it is possible to represent M:M indirectly, this make the model complex.

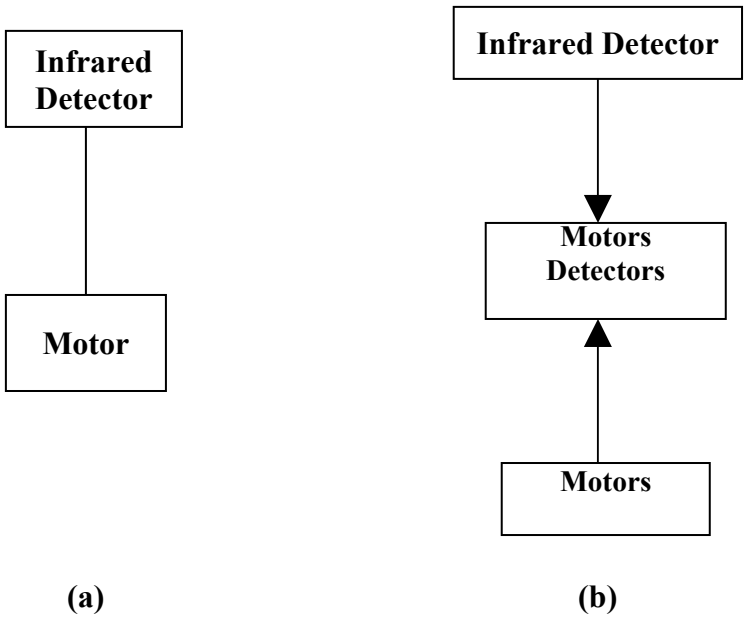


Fig 7-4: (a) many to many representation disallowed. (b) Intension representation for many to many relationship.

3. Record type must have one or more primary keys.

4. Explicit constraints on the domain and value of the attribute; for example Voltage is greater than 4 volt.

5. Membership, it can be Time Independent or Time Dependent membership.

a. Time Independent membership: it specifies the permanence of the connection between owner and member, the types of this property are:

1. Fixed set membership: a record has become a member of a set; it cannot be disconnected or moved to another set.

Example: user cannot move or disconnect Infrared Detector 11 which receives from Emitter 15.

2. Mandatory set membership: a record has become a member of a set; it cannot be disconnected but it can be moved to another set.

Example: user can move detector 13 (which controls motor 24) to control motor 22, but user cannot delete detector 13.

3. Optional set membership: a member can be moved or disconnected.

Example: Motor 21 can be moved to another power cable; also this motor can be disconnected.

b. Time Dependent set membership: it specifies the mechanism of establishing a connection in a set. The types of this property are:

1. Automatic set membership: at the moment owner is created immediately the member created automatically.

Example: at the moment user adds Emitter, new detector code created automatically.

2. Manual set membership: a member is connected manually by the user when he adds new owner record.

Example: when adding new power cable, user may add new motor code manually, or may not add this motor because the new power cable may for other devices.

7.4 Operations on Network Models

The two basic operations which are navigation operations and specification operations will be discussed separately.

7.4.1 Navigation Operations

Two currencies must exist. The first currency indicates the table which user wants to browse. The second currency indicates the last table which had been scanned by the user.

Two types of navigation may be applied on Network Model:

a. Navigation Through Table: Here an example shows how to browse voltages in power Supply table (the statements written in COBOL DML: data manipulation language):

- FIND Power Supply RECORD //place the currency on power supply table
- OBTAIN NEXT Voltage // browse the Voltage entries in Power Supply table

If the user stopped scanning, currency indicator stay on this table.

b. Navigation through Links: for example, suppose user wants navigate all emitters supplied by power cables exist in power supply table.

Figure 7-5 shows two related tables which are Power Supply table and Infrared Emitter table. There is no Power Cable 30 in Power Supply table. Then in "Emitter Supplied" link there is Emitter 17 supplied by Power Cable 30. So Emitter 17 doesn't appear in output.

User writes these statements to navigate Emitters supplied by power cables exist in power supply table.

- FIND Power Supply RECORD
- OBTAIN NEXT Emitter Code RECORD

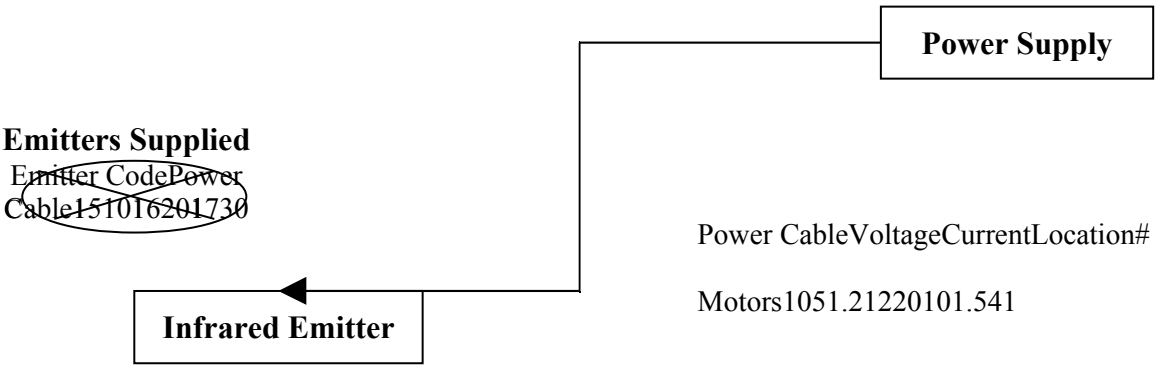


Fig 7-5: Two tables with one link (Emitters Supplied)

The output of these statements are:

Emitter Code
15
16

Fig 7-6: output of the navigation operation in the example

Actually this happened only when the user assigned ***Optional*** link between the two tables as mentioned in the previous section.

7.4.2 Specification Operations

Entries can be selected in one of the two ways:

1. According to Boolean expression (>, <, =, …) of criteria on the attributes of table. For example we will select power cables which provides voltage equal or greater than 10, the following statements do that (by NUL language):
    - S1 ← Power Cable
    - WHERE Voltage ≥ 10

Figure 7-6 shows the selection operation S1:

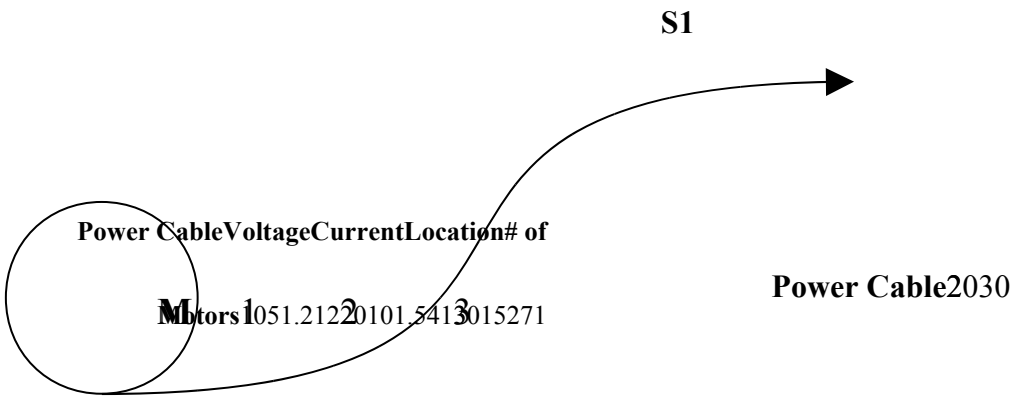


Fig 7-6: Selection operation S1

2. According to its relationship with previously selection. The previous example named the selection S1. User can build upon this selection.

Example: The previous selection S1 outputs Power cables 20 and 30.

The engineer wants to know which motors provided by these cables.

- S1← For S2
- BY Supplied Motors
- Motors

Figure 7-7 explain this compound selection S2. At first, program refer to S1 selection. Then, it makes another selection S2 which depends on the previous selection S1.

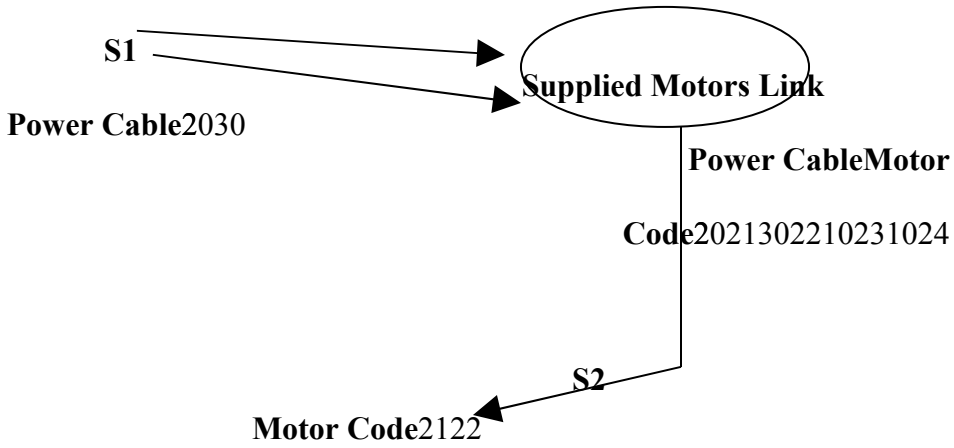


Fig 7-7: Specification operation S2 Built in Selection S1

7.5 Conclusion

Structures of the Network Model are simple; it contains the entities and relationships between them. This property serves complex scientific models, because it needs a simple model to be understood by users.

Assigning owner and member for each relation in network models helps scientific models to be strongly formed and integrated.

Network model suffers from difficulties in expressing many to many relationships. Another disadvantage is that network model disallows representing recursive links.

Conclusion

Managing scientific database utilizing a mediated warehouse architecture that provide a consistent interface to scientific data. It reduces the overall storage requirements of the warehouse, while maintaining access to all available data. Furthermore, our unique view of inter-database correspondences and extensive use of meta-data differentiate this approach from others while providing a

significant reduction in maintenance costs. It presents a general-purpose approach to managing scientific data, allowing scientists to better utilize the data they have worked so hard to produce.

The data which describes the system must be computed. Computing this scientific data requires a database design. So database theories used to rearrange scientific data.

A new method that manipulate data related to any system has been developed so data can be computed.

Using the proposed database model for the scientific system, a system analyze became easier and faster and a better understanding of sequence patterns, application structures, and the complex relationships between them will make structure prediction feasible.

Research in this area is based on the hope that in future when buying any machine we get with it a database software that can be used to understand the system and help in analyzing and maintenance.

Appendix A

EF Codd's 12

Database Rules

EF Codd's 12 Database Rules

The relational data model was first developed by Dr. E.F. Codd, an IBM researcher, in 1970. In 1985, Dr. Codd published a list of 12 rules that concisely defined an ideal relational database. These rules have been used as a guideline for the design of all relational database systems since then.

▲Codd's Rule #**1. Data is Presented in Tables**

- A set of related tables forms a database and all data is represented as tables; the data can be viewed in no other way
- A ***table*** (a.k.a. relation or entity) is a logical grouping of related data in tabular form (rows and columns)
- Each ***row*** (a.k.a. record or tuple) describes an item (person, place or thing) and each row contains information about a single item in the table
- Each ***column*** (a.k.a. field or attribute) describes a single characteristic about an item
- Each ***value*** (datum) is defined by the intersection of a row and column
- Data is ***atomic***; there is no more than one value associated with the intersection of a row and column
- There is ***no hierarchical ranking of tables***
- The relationships among tables are logical; there are ***no physical relationships among tables***

▲Codd's Rule #**2. Data is Logically Accessible**

- A relational database does not reference data by physical location; there is no such thing as the 'fifth row in the customers table'

- ***Each piece of data must be logically accessible by referencing 1) a table; 2) a primary or unique key value; and 3) a column***
- EXAMPLE. A specific employee in the' Employee' table (e.g., Mohammad ALi) and related information (last name, first name, ID, phone number, salary, etc) constitute a *row*. An employee's last name is a *column*. The name 'Doe' is a data *value*. Mohammad ALi's last name can be precisely located by referencing the 'employee' table, the appropriate column (last_name) and a unique key value (employee_id)

▲Codd's Rule #**3. Nulls are Treated Uniformly As Unknown**

- ***Null*** must always be interpreted as an ***unknown value***
- Null means no value has been entered; the value is not known
- 'Unknown' is *not* the same thing as an empty string ("") or zero
- EXAMPLE. If you pick up an item in a store and the price is not marked, the price is unknown (NULL); it is not free

- If not handled properly, nulls can cause confusion in your database
- EXAMPLE. If you search for all of the authors whose home City is not Tulkarem, the results will *not* include any authors with NULL in the 'City' column. SQL is very literal. You asked for authors where the City was NOT Tulkarem and NULL means 'unknown.' A NULL value for 'City' may mean that the City is Tulkarem and it may mean that it is not Tulkarem; you just don't know. Because the database engine can't tell for sure whether the City is not Tulkarem, a record with NULL will *not* be returned
- Nulls propagate through arithmetic expressions (e.g., 2 + NULL = NULL)
- Comparing a null to any value, including itself, returns NULL

▲Codd's Rule #**4. Database is Self-Describing**

- In addition to user data, a relational database contains data about itself

- There are two types of tables in a RDBMS: ***user tables*** that contain the 'working' data and ***system tables*** contain data about the database structure
- ***Metadata*** is data that describes the structure of the database itself and includes object definitions (tables, indexes, stored procedures, etc.) and how they relate to each other
- The collection of system tables is also referred to as the ***system catalog*** or ***data dictionary***
- System tables can be accessed in the same manner as user tables

▲Codd's Rule #**5. A Single Language is Used to Communicate with the Database Management System**

- There must be a single language that handles all communication with the database management system
- The language must support relational operations with respect to: ***data modification*** (i.e., SELECT, INSERT, UPDATE, DELETE), ***data definition*** (i.e., CREATE, ALTER, DROP) and ***administration*** (i.e., GRANT, REVOKE, DENY, BACKUP, RESTORE)

- ***Structured Query Language (SQL)*** is the de facto standard for a relational database language
- SQL is a '***nonprocedural***' or '***declarative***' language; it allows users to express what they want from the RDBMS without specifying the details about where it's located or how to get it

▲Codd's Rule #**6. Provides Alternatives for Viewing Data**

- A relational database must not be limited to source tables when presenting data to the user
- ***Views*** are ***virtual tables*** or abstractions of the source tables
- A view is an alternative way of looking at data from one or more tables
- A view definition does not duplicate data; a view is *not* a copy of the data in the source tables
- Once created, a view can be manipulated in the same way as a source table
- If you change data in a view, you are changing the underlying data in the source table (although there are limits on how data can be modified from a view)

- Views allow the creation of ‘custom tables’ that are tailored to special needs
- EXAMPLE: By not including the columns with sensitive information in a view definition, a view can be used to restrict a user’s access to the data
- Views can be used to simplify data access by ‘predefining’ complex joins; the concept is similar to that of a 'saved query'

▲Codd's Rule #**7. Supports Set-Based or Relational Operations**

- Rows are treated as ***sets*** for data manipulation operations (SELECT, INSERT, UPDATE, DELETE)
- A relational database must support ***basic relational algebra operations*** (selection, projection; & join) and ***set operations*** (union, intersection, division, and difference)
- Set operations and relational algebra are used to operate on 'relations' (tables) to produce other relations
- A database that supports only row-at-a-time (navigational) operations does not meet this requirement and is not considered 'relational'

▲Codd's Rule #**8. Physical Data Independence**

- Applications that access data in a relational database must be unaffected by changes in the way the data is physically stored (i.e., the physical structure)
- EXAMPLE: The code in an application that accesses data in a ***file-based database*** typically depends on the file format (e.g., the code references a 'phone number' field that is 10 characters wide, is preceded by the 'zip code' field, followed by the 'fax number' field...). If the layout of the data in the file is changed, the application must also be changed. In contrast, the storage and access methods (physical) used in a ***relational database*** can change without affecting the user or application's ability to work with the data. The user still only sees tables (logical structure)
- An application that accesses data in a relational database contains only a basic definition of the data (data type and length); it does not need to know how the data is physically stored or accessed

▲Codd's Rule #**9. Logical Data Independence**

- Logical independence means the ***relationships among tables*** can change without impairing the function of applications and ad hoc queries
- The database schema or structure of tables and relationships (logical) can change without having to re-create the database or the applications that use it

▲Codd's Rule #**10. Data Integrity Is a Function of the DBMS**

- In order to be considered relational, data integrity must be an internal function of the *DBMS*; not the *application program*
- ***Data integrity*** means the consistency and accuracy of the data in the database (i.e., keeping the garbage out of the database)
- There are three types of data integrity: entity, domain, and referential
- Within the database, data integrity can be enforced procedurally or declaratively

- ***Declarative data integrity*** involves placing or 'declaring' constraints on columns
- ***Procedural data integrity*** is maintained through code (i.e., through stored procedures or triggers)

▲Codd's Rule #**11. Supports Distributed Operations**

- Data in a relational database can be stored centrally or distributed
- Users can join data from tables on different servers (distributed queries) and from other relational databases (heterogeneous queries)
- Data integrity must be maintained regardless of the number of copies of data and where it resides

▲Codd's Rule #**12. Data Integrity Cannot be Subverted**

- There cannot be other paths into the database that subvert data integrity; in other words, you can't get in the 'back door' and change the data in such a manner as data integrity is violated
- The DBMS must prevent data from being modified by machine language intervention

Appendix B

R Series RAW Power Supply Manual

References

1. Gijs Van Tulder. 2003. *Storing Hierarchical Data in a Database.* www.sitepoint.com.
2. Jennifer Cartier,John Rudolph,and Jim Stewart. 2001. *The Nature and Structure of Scientific Models*. The National Center for Improving Student Learning and Achievement in Mathematics and Science (NCISLA).
3. Roman Frigg and Stephan Hartmann. 2005. ***Scientific Models.***
4. Alqadi, Ziad and Abdelraheem Albashiti. 1997. *Data III Plus*. Amman: Dara Alsafa'.
5. Bloor, Robin. 2004. *The Failure of Relational Database*. Baroudi Bloor International Inc: http://www.baroudi.com.
6. Boudjlidi, Nacer. 2003. *Database Systems: Theory and Application*. Halap-Syria: Sho'a' for publishing and science.

7. Chen, Peter P. 2005. *Entity-Relationship Modeling: Hstorical Events, Future Trends, and Lessons Learned*. Louisiana State University: http://bit.csc.lsu.edu.
8. Elmasri and Navathe. 2004. *Fundamentals of Database Systems*. Georgia, College of Computing: www-static.cc.gatech.edu.
9. Fehmi, Mostafa A. Alwahab, Jmal Abdelmo'ti, Abdelaziz Alhareri, Ezzat Shaddad and Ala' Eddin Sharqawi. 1993. *Computer and Databases*. Egypt: Delta Books Group.
10. Floyd, Thomas L. 1999. *Electronic Devices*. New Jersey: Prentice-Hall Inc.
11. Frick, David R. 2004. *Characteristics of a Relational Database*. http://www.frick-cpa.com.
12. Hernandez, Michael J. 2002. *Data Design Tips*. http://www.datatexcg.com.
13. Howe, David. 2001. *Data Analysis for Database Design*. England: Butterworth-Heinemann.
14. Inmon, Bill. 2001. *A Brief History of Database Design*. DM Review Magazine: http://www.dmreview.com.

15. Matko, Richard K. and Borut Zupancic. 1992. *Simulation And Modeling Of Continuous Systems*. USA: Prentice Hall.

16. Pedersen, Alf A. 2004. *Development Cycles: Entity Relationship Modeling*. Database Design Resource Web Site: http://www.databasedesign-resource.com.

17. Tsichritzis, Dionysios C. and Frederick H. Lochovsky. 1982. *Data Models*. New Jersey: Prentice - Hall.

18. T. Critchlow, K. Fidelis, M. Ganesh, R. Musick, T. Slezak. *Data Foundry : Information Management for Scientific Data*. U.S. DOE by LLNL.