DE-MYSTIFYING DATA BASE NORMALIZATION

Patrick C. Witiw and Gil Harrison

Brant Computer Services Limited
9637A-45 Avenue
Edmonton, Alberta
Canada

(403) 438-9123

Preface

On a beautiful autumn day in Edmonton in October 1985, two consultants from a computer services company spoke to a client's analysts and programmers on the subject of the performance of a certain 4th GL upon IMAGE data bases on the HP3000 Series 48.

Following a presentation on the internals of and the interaction between MPE, IMAGE and the 4th GL a pre-lunch question from the audience was asked that the speakers were not prepared for:

> "What effect does the normalization of a base to the 3rd normal form have on performance compared to that of the 1st and 2nd normal form databases?"

The speakers were only barely familiar with the term data base normalization and had to admit, with some embarassment, that they had no working knowledge of the associated techniques. They also made a mental note to investigate normalization soon.

During the next two years, the consultants were involved with the design, implementation and maintenance of a number of IMAGE data bases but never got around to learning about and applying data base normalization.

Finally, the time has come for my collegue Patrick C. Witiw and I to set forth and de-mystify data base normalization.

Gil Harrison

Introduction

Data administration today uses a structured approach to managing a company's resources. The decision makers have access to information, when needed. This is accomplished by combining the efforts of both the data processing people and the end users to identify, define and implement the data bases which contain the company's whole data resource. In doing this the two groups establish the overall logic for the company's data infrastructure. Data bases are now designed and implemented as common data bases and are used by MIS built systems and end user built systems.

There are several major design techniques available for building data bases and this paper will serve to explain one technique call data normalization which is a popular "buzzword" in the vocabulary of data base designers and analysts. At times it seems these specialists use the term normalization to justify their existence to the less informed among us. Though normalization of data is derived from theoretical mathematics and is in the repertoire of the skilled data base designer, we will demonstrate how normalization is a technique by which anyone who understands the functional aspects of one's data can produce data structures that rival those produced by the elite of the data analysis profession.

What is normalization and when do we use it?

Normalization is a technique for decomposing data into smaller structures in which each field is totally dependent upon the primary key of the entity in which it resides. This theory was designed by E. F. Codd who is recognized as the father of data base normalization. Codd translated the origin of normalization theory from abstract-theoretical mathematics into a process which is more human.

An important factor to note about data normalization is that it is not new! Data Base Administrators (DBA) have been intuitively normalizing data for years. The only thing new about the subject is that we are starting to realize that normalization should not be an exclusive skill of the DBA. R. C. Perkinson, a 1980's data base structure and design guru, reinforces this by writing:

> "Normalization depends on a knowledge and understanding of the data in the functional business unit being examined

and the way it relates together. It does not depend upon any particular data base knowledge or skill." (1)

If you are an analyst, or an end user, or even in a position of management and you understand the flow of your data, then by using Codd's process step by step, you should be able to produce a normalized data structure comparable to one produced by an experienced DBA. And, if I were to ask you your reasons for your entity structure and then ask the DBA the same question I would receive two completely different answers. The DBA would say "I based it on experience". You would give me definate reasons for your structure because you understand how the data is used and you would use normalization to express your understanding. Gane and Sarson, a husband and wife team who specialize in data base design, submit that the normalization process is "inspired common sense".

Without looking very hard we can see normalization as it is applied in our day to day lives. Take, for example, McDonald's restaurants. How successful would the organization be if all they offered was beef, chicken, bacon, fish, potatoes and eggs mixed all together in one combination called McStew? Obviously they would not be where they are today because:

1) The prospect of all the food mixed together is unappealing and would not go over very well with their major end user, the junk food addict,

2) The prospect for accessing what you want quickly is quashed when you have to spend valuable time separating the fries you have craved from the rest of the goo,

3) Too much money would be wasted by the end user eating only the craved fries and throwing the the rest of the goo away.

How about the way you organize your clothes dresser. Do you have your socks, undergarments and sweaters mixed all together through out your drawers? Or do you have them separated in their own respective drawer for quick and easy accessibility? Starting to get the picture? The list of "inspired common sense" is endless. Normalization is the simple concept of breaking down large views of data into simple structures (SIC).

So what exactly is the normalization process?

Here is how it sounds when the DBAs talk: There are three main steps to normalization and each step has its own specific rule. The three steps are:

1) FIRST NORMAL FORM
   - repeating groups are moved into a new entity

2) SECOND NORMAL FORM
   - attributes that are wholly dependent on only part of a primary key or primary compound key are moved into a new entity

3) THIRD NORMAL FORM
   - attributes wholly dependent upon another key within an entity are moved into a new entity

One major problem in developing data bases using the normalization technique is understanding the lexicon of normalization and mathematical terms. We eliminate this problem by eliminating the jargon and replacing it with common everyday terms. However, here are some terms that we cannot avoid and therefore must be defined before we can continue:

ENTITY
- a record containing one or more data fields

KEY
- a data field or combination of data fields used to identify and locate a record

PRIMARY KEY
- a key that is used to uniquely identify a record (eg: a TELEPHONE NUMBER is unique to each household)

SECONDARY KEY
- a key that does not uniquely identify a record; that is, more than one record can have the same key value (eg: an AREA CODE is related to many household TELEPHONE NUMBERs)

During our research of normalization we became quite familiar with the Customer Order Processing (COP) application. This was due to the fact that all the authors of our research material (Perkinson, Martin, Gane and Sarson) used the COP system as an aid for their interpretation and explanation of the normalization process. We, on the other hand, are going to illustrate normalization by examining the development of a Tape Archive and Retrieval System (TARS). TARS is a functioning system that we developed to run on the HP3000, in early 1987. The system was originally programmed using COBOLII but later was converted into a PowerHouse application. The TARS data base makes use of IMAGE, Hewlett-Packard's hierarchical data base management system.

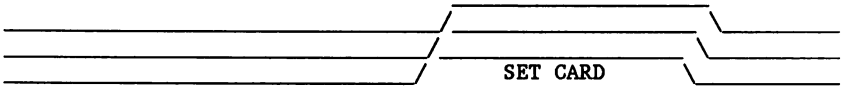
When we first developed the TARS application we used our experience in IMAGE design to build the data base. At the time we did not have a clear understanding of the normalization process. It was only after our research for this paper that we decided to apply the normalization techniques to TARS. We wanted to prove how easy normalization could be.

In doing this our first task was to break down and reorganize the TARS data base fields in order to produce on paper a list which represented the "un-normalized state". We then proceeded to apply the three rules of normalization to the list. The result was the creation of a normalized data base which looked almost identical to the original data base. The differences will be explained later.

Before starting the normalizing process we must develop an understanding of how a user will use the TARS information. In order to gain this understanding it is best that we look at the manual tape archive and retrieval system which was replaced by TARS.

By making use of a file drawer containing index cards called SET CARDS, the manual tape archive and retrieval system provided a Computer Operations department with the ability to keep an orderly account of all magnetic tapes. On these SET CARDS we wrote information pertaining to each set of tapes in our library. There was one SET CARD per tape set. A unique ACCOUNT NAME was assigned to each user client and inhouse department. We catalogued the SET CARDs by account. When an operator was asked to locate a set of tapes he would ask the requesting party for the ACCOUNT NAME to which the tapes belonged, the DESCRIPTION of the tape set's contents and the tape set creation date. Given this information the operator would review all the SET CARDS in

the file drawer for the given ACCOUNT NAME. After finding the appropriate SET CARD the operator would use the information on the card to locate the requested tape set. All tapes belonging to a given tape set are stored together on a rack or in a box with the SET NUMBER prominently displayed. The tape number on the band of the tapes would correspond to the VOLUME NUMBERs listed on the SET CARD. Figure 1A is an example of a SET CARD which was used in our manual tape archive and retrieval system.



SET NUMBER 0134 ACCOUNT NAME ACRC

ACCOUNT DESCRIPTION ACCOUNTS RECEIVABLE

SET DESCRIPTION

5/70 MARCH 1987 Version of the INTEREST data base

SET STORAGE Brant Edm. Tape rack # 4 SET CREATION DATE Jan. 14/86

SET SEQ/TOT 4 SET EXPIRY DATE Jan. 14/88

VOLUME INFORMATION

VOLUME NO	LENGTH	SEQ/NUM	INIT/DT	CLEAN/DT	NUM/CL	MANUF
BE1012	2400	1	Jan. 14/86		ø	3m Black Watch
BE1013	2400	2	Jan. 14/86		ø	3m Black Watch
BCA659	2400	4	Sept 4/83	June 4/85	1	Memorex
AE101	3600	3			ø	Memorex

FIGURE 1A

A small to medium sized operations department can have in circulation anywhere from 500 to 1500 tapes. This is a significant amount of tapes especially if you are the computer operator who is given the task of manually locating a set of archived audit tapes requested by the Accounts Receivable department. We developed TARS as a solution to simplify manual tasks of this nature. TARS is an an online application and will help you to easily and readily find a requested set of tapes. The requested information can be presented to the user in either display terminal format or paper report format. Access to TARS information is done via one of three keys as follows.

1) A unique SET NUMBER represents a specific sequence of related tapes,

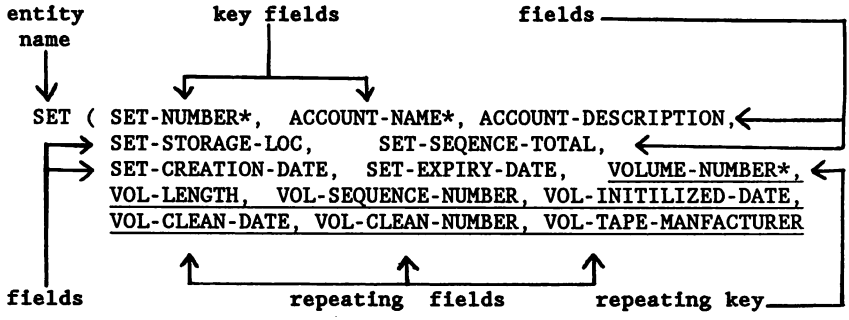
2) A unique VOLUME NUMBER represents one magnetic tape,

3) An ACCOUNT NAME represents a group of related SET NUMBERS.

Now that we understand the function of the data we can proceed to normalize it.

Initial Process

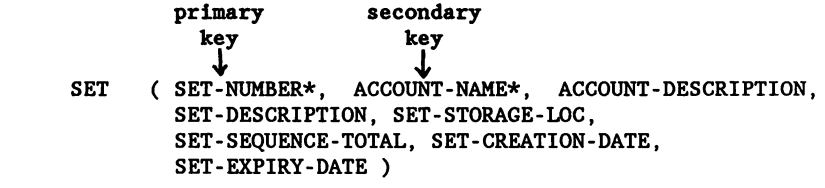
You will recall that a key field is used to identify and locate records. Having determined the keys, we can now create the preliminary order list which will contain all the fields on the SET CARD. The preliminary list represents the "unnormalized state" and the order in which the data is listed is important for ease of analysis. List the entity name first, the keys next, then list the rest of the fields. "*" denotes key.

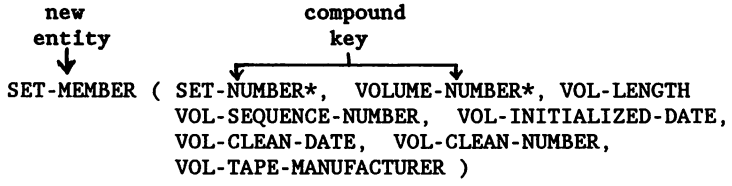


Fields that can take on more than one value occurence, such as VOLUME-NUMBER and the information pertaining to VOLUME-NUMBER should be underlined. VOLUME-NUMBER is now known as a repeating key. We are now ready to start the normalization of the SET CARD data.

First Normal Form

As indicated earlier, this step requires us to identify and remove the repeating groups of fields that exist in the entity called SET. This is an easy task since we already indicated the repeating groups by underlining them. So we move VOLUME-NUMBER, VOL-LENGTH, VOL-SEQUENCE-NUMBER, VOL-INITIALIZED-DATE, VOL-CLEANING-DATE, VOL-CLEAN-NUMBER AND VOL-TAPE-MANUFACTURER from the SET entity and into a new entity called SET-MEMBER.





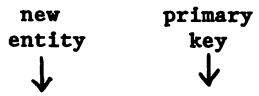
Removing repeating groups always results in the newly created entity(s) having a primary compound key. In this application the entity SET-MEMBER's primary compound key is SET-NUMBER + VOLUME-NUMBER. The key SET-NUMBER is a unique key in the entity SET and will now be referred to as a primary key. Whereas ACCOUNT-NAME which is not a unique key in the entity SET will be referred to as a secondary key. Labeling keys, as you see, requires an understanding of the data.

Second Normal Form

This step requires us to identify and remove the fields which are wholly dependent on part of the primary key or primary compound key. In the entity SET-MEMBER all of the non-key fields are wholly dependent on VOLUME-NUMBER and VOLUME-NUMBER is part of the primary compound key. So we copy VOLUME-NUMBER and move VOL-LENGTH, VOL-SEQUENCE-NUMBER, VOL-INITIALZED-DATE, VOL-CLEAN-DATE, VOL-CLEAN-NUMBER AND VOL-TAPE-MANUFACTURER into a new entity called VOLUME shown as follows:

```
SET ( SET-NUMBER*, ACCOUNT-NAME*, ACCOUNT-DESCRIPTION,
      SET-DECRIPTION, SET-STORAGE-LOC, SET-SEQUENCE-TOTAL,
      SET-CREATION-DATE, SET-EXPIRY-DATE )

SET-MEMBER ( SET-NUMBER*, VOLUME-NUMBER* )
```




```
VOLUME ( VOLUME-NUMBER*, VOL-LENGTH, VOL-SEQUENCE-NUMBER,
         VOL-INITIALIZE-DATE, VOL-CLEAN-DATE,
         VOL-CLEAN-NUMBER, VOL-TAPE-MANUFACTURER )
```

VOLUME-NUMBER becomes the primary key of the entity VOLUME. The entity SET does not contain a primary compound key but rather a primary key which is SET-NUMBER. If SET-NUMBER was made up of two sub-fields whereby the first sub-field was composed of two digits which represented a company department code and there were fields in the entity SET which were dependent on the department code then we would perform the second normal form analysis on this entity. However, in the TARS application SET-NUMBER is used as a single field and no dependencies exist on only part of the SET-NUMBER key, therefore the second normal form is not performed on the entity SET.

Third Normal Form

Our last step in the normalization process is to identify and remove the fields which are wholly dependent upon another key or field within the entity in which they reside. The difference between the second normal form and the third normal form is the second normal form deals with dependancies only on primary keys whereas the third normal form deals with dependencies on all keys and fields except primary keys. If an entity in the second normal form has all non-key fields wholly dependent on only the primary key or primary compound key it is then considered to be in the third normal form. The third normal form process is one in which we remove fields seemingly in the "wrong" entity.

Returning to our example, we recognize that the entity SET has a key which is not a primary key or primary compound key. A perfect target for the third normal form analysis. With further examination we see that the non-key field ACCOUNT-DESCRIPTION is wholly dependent on the secondary key ACCOUNT-NAME. As a result, we copy the secondary key ACCOUNT-NAME and move ACCOUNT-DESCRIPTION into a new entity called ACCOUNT shown as follows:

```
       SET ( SET-NUMBER*, ACCOUNT-NAME*, SET-DESCRIPTION,
             SET-STORAGE-LOC, SET-SEQUENCE-TOTAL,
             SET-CREATION-DATE, SET-EXPIRY-DATE )

    new        primary
  entity         key
    ↓             ↓
  ACCOUNT ( ACCOUNT-NAME*, ACCOUNT-DESCRIPTION )

SET-MEMBER ( SET-NUMBER*, VOLUME-NUMBER* )

    VOLUME ( VOLUME-NUMBER*, VOL-LENGTH, VOL-SEQUNCE-NUM,
             VOL-INIT-DATE, VOL-CLEAN-DATE, VOL-CLEAN-NUMBER
             VOL-TAPE-MANUFACTURER )
```

The entity VOLUME is an example of an entity in the second normal form and having all of its non-key fields wholly dependent on only the primary key, thus qualifying it as a third normal form entity.

The result of the third normal form process is to produce entities whose non-key fields are independent from each other but still dependent on the primary key or primary compound key of the entity in which they reside.

We have just completed what we set out to do! With relative ease we systematically normalized to the third form the TARS application data. To restate our point we ask: "what's the big deal about normalization?" A blunt question maybe - but it is necessary to realize that normalization is a simple technique of structured intuition which helps us to organize data base data.

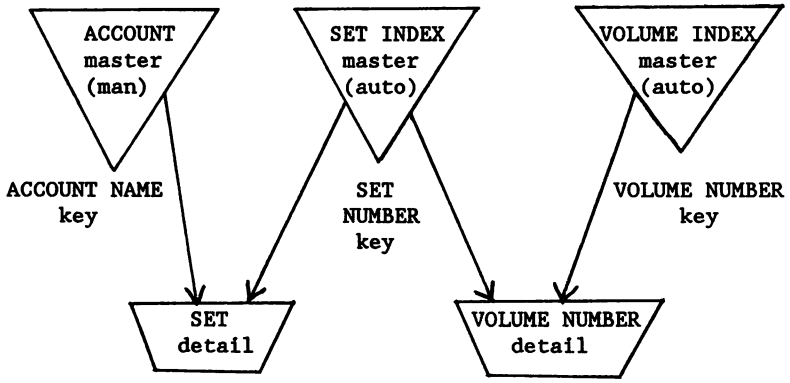
Now we are ready to implement the logical design that our normalization produced. The physical enviroment for our data base is defined by the HP3000 product called IMAGE. It is important to realize that in moving our logical design to the physical environment of IMAGE the entities will be represented as data sets.

The entity SET will be represented by a detail data set. The entries (records) of the set will be accessible on SET-NUMBER and ACCOUNT-NAME via the IMAGE paths defined by the automatic master called SET-INDEX and the manual master called ACCOUNT.

The entity VOLUME will be represented by a detail data set. The entries of the set will be accessible on the VOLUME-NUMBER and the SET-NUMBER via the IMAGE paths defined by the automatic masters VOLUME-INDEX and SET-INDEX respectively.

The entity ACCOUNT will be represented by the manual master called ACCOUNT in which the entries are accessible by the search key ACCOUNT-NAME. This master will also serve as an access path though the detail called SET.

In our implementation there is no need for the logical design's SET-MEMBER entity because IMAGE access paths on the keys SET-NUMBER and VOLUME-NUMBER are provided though the two index data sets. The physical layout of the TARS data base is shown on the following page.



SET (SET-NUMBER*, ACCOUNT-NAME*, SET-DESCRIPTION,
SET-STORAGE-LOC, SET-SEQUENCE-TOTAL,
SET-CREATION-DATE, SET-EXPIRY-DATE)

ACCOUNT (ACCOUNT-NAME*, ACCOUNT-DESCRIPTION)

VOLUME (VOLUME-NUMBER*, SET-NUMBER*, VOL-LENGTH,
VOL-SEQUENCE-TOTAL, VOL-INITIALZE-DATE,
VOL-CLEAN-DATE, VOL-CLEAN-NUMBER,
VOL-TAPE-MANUFACTURER)

index
data set
↓
SET-INDEX (SET-NUMBER*)

index
data set
↓
VOLUME-INDEX (SET-NUMBER*)

These five data sets are exactly the same as found in the original data base designed without normalization. Through normalization we have proven our original design.

Conclusion

To summarize, normalization is a logical level development methodology which in the grand scheme of database design and implementation is only one piece of the pie. Logical development enables us to determine the relationships between the data fields and the entities in which they exist. The other pieces of the pie consist of the following:

1) The physical level of development expresses the logical relationships in terms of software and machine enviroments

2) The actual implementation of the data base

3) The performance analysis of the data base

The success of a data base application running smoothly and efficently depends largely on the physical design, the responsibility for which is in the domain of the DP professional. However, just because the data base application is considered to be successful by the physical designers does not guarantee approval from the end user. The end users will give their approval only when their needs are fulfilled. Involving them in the logical level of development is a key factor in developing a data structure which will be accepted by them. After all, data management is primarily a business function and the decisions regarding the data can realistically be made only by the people who are the ultimate users.

Footnote

1 - R. C. Perkinson, Data Analysis: The Key to Data Base Design, chapter 4 page 41

Bibliography

1) Gillerson, Mark L. Database Step-By-Step. (John Wiley & Sons, Inc., 1985).

2) Martin, James. Managing the Database Environment. (Prentice-Hall Inc., 1983).

3) Perkinson, Richard C. Data Analysis: The Key to Data Base Design. (Lellesly, Mass: QED Information Science).

4) Turk, Thomas A. Planning and Designing the Database Environment. (Van Nestrand Reinhold Company Inc., 1985).

April, 1988