THE RELATIONAL VERSUS HIERARCHICAL DATABASE DEBATE CONTINUES
Alma Zuniga, SAS Institute Inc., Austin, Texas

ABSTRACT
The marketplace of database products has been shaped in large part by the leading academicians in computer science and, of course, by IBM. Once the relational technology came on the scene, the leading scholars were quick to embrace the idea. After all, hardware storage was getting cheaper and CPU power, once isolated in massive computer rooms, was making its way to the desktop. Is today's technology leading towards a relational model?

The purpose of this paper is to explain the difference between a relational and hierarchical data structure and present a description of the characteristics that define these structures. Although the data structure is a part of the database design, there should be other considerations when selecting the correct database management system.

So how do you choose and what tools are available to make these decisions? The SAS System provides an opportunity to take advantage of any data storage structure including flat files, SAS data sets, hierarchical SYSTEM 2000 databases, and relational databases, such as DB2 and SQL/DS.

INTRODUCTION
The following are some of the issues that should be considered when looking for a database management system. Why consider a database? What is involved in implementing a database application? What advantages would the user gain by using a database? And finally, how the data structure you choose, takes advantage of the tools available in Version 6 of the SAS System.

DEFINITIONS
Hierarchical Data Structure
In the hierarchical database model, trees are used to represent the logical structure of data. A tree consists of records connected by branches. The record at the top is called the root. Trees are distinguished from similar structures in that every record of a tree has exactly one parent (except for the root, which has no parent). The descendants of a record are called children. Records that have the same parent are called siblings.

Levels in Database Definition

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>101: POSITION WITHIN COMPANY</td>
<td>101: SALARY RELEVANT POSITION</td>
<td>101: EDUCATIONAL BACKGROUND</td>
</tr>
<tr>
<td>102: POSITION TITLE</td>
<td>112: EMPLOYEE RATE</td>
<td>102: HIGH SCHOOL</td>
</tr>
<tr>
<td>103: DEPARTMENT</td>
<td>113: EFFECTIVE DATE</td>
<td>102: MIDDLE SCHOOL</td>
</tr>
<tr>
<td>104: MANAGER</td>
<td>114: CURRENT DURATION</td>
<td>102: UNIVERSITY</td>
</tr>
<tr>
<td>105: POSITION TYPE</td>
<td>106: END DATE</td>
<td>102: OTHERS</td>
</tr>
<tr>
<td>106: START DATE</td>
<td>115: POSITION TYPE</td>
<td></td>
</tr>
<tr>
<td>107: END DATE</td>
<td>120: JOB DESCRIPTION</td>
<td></td>
</tr>
<tr>
<td>108: EMPLOYEE NAME</td>
<td>121: EMPLOYEE POSITION</td>
<td></td>
</tr>
<tr>
<td>109: EMPLOYEE ID</td>
<td>122: EMPLOYEE ADDRESS</td>
<td></td>
</tr>
<tr>
<td>110: SALARY WITHIN POSITION</td>
<td>123: EMPLOYEE PHONE</td>
<td></td>
</tr>
<tr>
<td>111: POSITION TYPE</td>
<td>124: EMPLOYEE EMAIL</td>
<td></td>
</tr>
<tr>
<td>112: EMPLOYEE RATE</td>
<td>125: EMPLOYEE SOCIAL SECURITY NUMBER</td>
<td></td>
</tr>
<tr>
<td>113: EFFECTIVE DATE</td>
<td>126: EMPLOYEE HOME ADDRESS</td>
<td></td>
</tr>
<tr>
<td>114: CURRENT DURATION</td>
<td>127: EMPLOYEE WORK ADDRESS</td>
<td></td>
</tr>
<tr>
<td>115: POSITION TYPE</td>
<td>128: EMPLOYEE TITLE</td>
<td></td>
</tr>
<tr>
<td>116: EMPLOYEE ID</td>
<td>129: EMPLOYEE SALARY</td>
<td></td>
</tr>
<tr>
<td>117: EMPLOYEE NAME</td>
<td>130: EMPLOYEE TITLE</td>
<td></td>
</tr>
</tbody>
</table>

Relational Data Structure
In the relational model, tables are used to represent the logical structure of the data. A table consists of a row of column headings together with zero or more rows of data values. For a given table, (a) the column heading row specifies one or more columns; (b) each data row contains exactly one value for each of the columns specified in the column heading row. Every table should have a primary key - that is, a field, or field combination, that serves as a unique identifier for the records in that table. Primary keys should not accept null values.

S | SF | SPNAME | STATUS | QTY
---|----|--------|--------|-----
S1 | S1 | SMITH | 20 | LONDON
S2 | S2 | JONES | 10 | PARIS
S3 | S3 | BLAKE | 30 | PARIS
S4 | S4 | CLARK | 20 | LONDON
S5 | S5 | ADAMS | 30 | ATHENS

P | P# | PNAME | COLOR | WEIGHT
---|----|--------|-------|------
P1 | P1 | NUT | RED | 12
P2 | P2 | BOLT | GREEN | 17
P3 | P3 | SCREW | BLUE | 17
P4 | P4 | SCREW | RED | 14
P5 | P5 | CAM | BLUE | 12
P6 | P6 | COG | RED | 19

The suppliers-and-parts database (relational version)

417
CHARACTERISTICS

When asked to list the characteristics of a hierarchical database, the following are widely used:

- parent and child relationships
- one to many relationships
- predefined data structure
- limitable data redundancy

When asked to list the characteristics of a relational database, the following are widely used:

- table oriented (rows and columns)
- no predefined data structure
- common fields between files (primary keys)
- ability to build reports accessing several databases
- allows use of SQL
- requires more storage space

Let's consider some data structure issues:

1. Does a relational database imply that there is no predefined structure?

   It is a very common assumption that database design is unnecessary in a relational system. It presumably stems from the assumption that (a) the structure of an individual relation is extremely simple, and (b) the logical structure of an entire relational database is thought to be quite simple. With those assumptions in mind the user assumes that a relational database is structure-free. But the fact remains that there certainly is a structure, and that structure certainly does have to be designed. There is no question that the logical design can still be complex if the database itself is complex. When designing a database through either data structure the business applications needs should be considered. You should be able to apply changes to the design of the database as your user requirements change.

2. Can SQL be used to access your data?

   This can be applied to both database structures. The debate no longer is between data structures but between tools available to access the databases. In this case it is necessary to evaluate the features that are available through various software vendors. In fact, in Version 6 of the SAS System, you can now use PROC SQL to interface with SYSTEM 2000 as well as other database management systems.

3. Does a relational database use more storage space than a hierarchical database?

   A relational database could use exactly the same physical storage structure as one of today's hierarchical databases, in which case the storage requirements would be exactly the same. However, in most current relational systems, relational tables map into uncompressed stored files, and those files in turn are heavily indexed. These files tend to require more space for either or both reasons: (a) They provide more entry points into the data (equivalently, more access paths), and (b) data fields often take up more space than pointers.

4. Do relational databases contain large amounts of data redundancy?

   "Redundancy" within relational data structures, often times, exists because of "all those keys." In a relational structure every table consists of a primary key which can be used to link more than one table at a time. Thus, for each additional key, more storage requirements are necessary.

5. Do hierarchical databases have less redundant data?

   Within hierarchical data structures, relationships are established by arranging data into records and records are positioned into certain levels within the structure. The levels are achieved by ranking items with values that occur once in the first record (root). Items within records having a one-to-many relationship with other record items rank higher in the database hierarchy. Ranking the items in this way reduces repeated values and gives you a clear picture of your data. A hierarchical database will not eliminate redundancy but if the database is designed appropriately, it can reduce redundant data values considerably.

6. Are hierarchical structures more powerful than relational structures?

   All data can fit into either the hierarchical or relation format. Both data structures consist of data stored in records. Data is accessed differently in each structure. The hierarchical database has links predefined that are visible to the user, whereas, the relational database consists of tables that have primary keys used to link the data. The power of the data structure relies on the tools made available to access the data in a database.

   Note: Is it easier for a user to understand the flow of the data based on primary keys or would a tree structure better define the flow?

THE REAL DEBATE

Some issues have been pointed out to indicate that a database structure is not the only consideration when looking for a database management system. The real debate is not the database structure but the tools available to access your data. Let's consider why we would need a database management system and its advantages.
Why a Database?

Why should a company choose to store its operational data in an integrated database? It provides the company with centralized control of its operational data, accurate up-to-date information, fast data retrieval and updating capabilities.

When you set up a database application three things are involved:

- **The User or Application programs**

  The user, directly or indirectly via application programs, operates with the database system in three modes: definition, access and manipulation.

  **Definition:**
  
The user defines the database by specifying the formats of the data and the relationships among data items.

  **Access:**
  
The user accesses data by requesting the database system to retrieve portions of the database. This occurs in two steps. First, the required data must be found. The user may have several options to request the data. To the user, they are equivalent. To the database system, one option may result in easy access and efficient processing while another may be very slow and cause repetitive, wasteful input/output processing. Secondly, once the data is found, it must be presented to the user in a familiar and useful format.

  **Maintenance:**
  
The user maintains the database by modifying or updating it. Data can be added, deleted, or changed. The user may have a requirement to modify not only the data but also the database structure. Database maintenance is extremely important. The database is almost useless if it does not contain up-to-date, valid data. If it is difficult for the user to maintain the database, modifications may not be made on a timely basis, resulting in incorrect and possibly damaging information.

- **The Database System**

  The database system is the interface between the user of application programs and the database. It is a set of computer programs that translates the user's commands into actions performed on the database. The functions of a database system are to translate, transform and transfer. It must translate the user's requests to determine the desired database operations. Then it must transform the requests into operations to be performed on the database files. Finally, the data must be transferred between main memory and secondary storage devices to the user.

- **The Database**

  The third component of a database processing application is the database itself. It is the collection of data necessary to satisfy the defined needs of the user. It contains not only data but also information about the structure of the data. This includes information about the format of the data as well as the relationships among data items.

What are some considerations when looking for the right database management system?

1. **The amount of redundancy in the stored data can be reduced.**

   In most current systems each application has its own private files. This can lead to considerable redundancy in stored data, resulting in wasted storage space. With central control, one can identify that two applications or more require the same data, and by integrating the data files one can store the data in one place that can be shared by the applications.

   Does the data structure you choose for your database reduce data redundancy? In some cases you can see the differences in storage savings. In a hierarchical database you can take advantage of the structure to reduce redundancy (not eliminate) within the database itself. In a relational database, as mentioned earlier, primary keys will contain duplicate data through the tables as linking values.

2. **Problems of inconsistency in the stored data can be avoided (to a certain extent).**

   The same rule applies to eliminating problems of inconsistency in the data stored. If two departments are using their own private files that consist of the same data and one department makes changes to their data and not to the other departments data. The data will no longer be consistent.

   Does the data structure you choose solve data inconsistency? No.

3. **Multiple access to your data.**

   One of the primary reasons for storing your data in a database is to be able to share the data among other users. With a database, data does not have to be stored redundantly to be used for multiple purposes. By storing your data in a database you get accurate, up-to-date information on demand at any time. Most database systems have “multi-user” aspects that allow for concurrent access - update and retrieval - with the software performing the task of coordinating to ensure data integrity.
Does the data structure you choose for your database affect the capability of sharing your data? No.

4. Indexing.

Indexing gives you faster access to your data. Looking something up in the database is rather like looking something up in a book. The system has two options: It can do a sequential scan or it can do an index lookup - assuming the data in question has an index. An index lookup will usually be faster than a sequential scan.

Does the data structure you choose for your database imply that you have index capability? No.

5. Security restrictions can be applied.

In a shared environment it is likely that not every user needs to see, or should be allowed to see, every piece of data in the database. A shared database will typically include information of varying levels of sensitivity, and different users will be authorized to access the portions of the database that they need to accomplish their own particular job.

Does the data structure you choose for your database affect the security of your data? No.

6. Data integrity can be maintained.

The problem of integrity is the problem of ensuring that the database contains only accurate data. Inconsistency between two entries representing the same fact is an example of lack of integrity. Centralized control of the database helps in avoiding some of these situations by defining validation procedures to be carried out whenever any storage operations is attempted.

Does the data structure you choose for your database affect the data integrity? No.

7. Data Independence.

This is a major objective of a database system. The application concerned should not depend on any one particular storage structure and access strategy. The database should be able to grow without affecting existing applications. It should be possible to extend an existing stored record type by the addition of a new field. These new field types will simply be invisible to all previous applications. The advantage of data independence is that it enables the user to concentrate on the logical structure of the data and ignore irrelevant physical details.

Does the data structure you choose for your database affect the data independance you gain from a database. No.

8. Recovery.

In the event of damage to any portion of the database - caused by human error, or failure in the hardware or supporting operating system. It is essential to be able to repair the data concerned with the minimum of delay and with as little effect as possible on the rest of the system.

Does the data structure you choose for your database imply that recovery is available. No.

Other options that must be considered when looking for a database management system are:

- Hardware platforms
- Expense of the database
- Ease of Use (tools provided by software to access data)
- Programming languages supported
- Fit for your business application
- Technical Support
- Education

CONCLUSION

In a relational or hierarchical database, the software allows you to organize and access data in a manner suited to your needs without having to be concerned with the physical storage of the data. The physical and logical structures are independent. However, the software uses both structures to store data in an efficient manner and to access data based on both the values and the relationships among the data. The database does not consist of a data structure alone. Data structures alone are not particularly interesting, or useful. It is not sufficient to compare databases by data structures, you need to consider the tools provided to access the data. To summarize the debate on relational versus hierarchical databases we need to consider several factors other than the database structure alone. In most cases you will be able to do the same operations in either data structure. The SAS System currently provides interfaces to most database management systems. This allows you to further hide the database structure from the end user. Engines are a new feature of Version 6 of the SAS System. The Multiple Engine Architecture is designed to meet several goals, including a transparent method of accessing data stored in the database management systems of other vendors. With such a method, it is no longer necessary to convert these external files into SAS data files before using the data in SAS applications or procedures.