white paper

Rocket

Nested Relational Databases

© Rocket Software, Inc. or its affiliates 1990 – 2014. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

Table of Contents

Table of Contents **1**

Introduction **2**

Abstract 2

Basic Relational Database Technology 3

Normalization of Relational Databases 4

Relationship Relations 7

Limits of 1NF Relational Databases 8

Eliminating the 1NF Constraint 9

Nested Relational Databases **9**

Nested Tables 10

Implicit Relationship Relations 12

Technical Advantages of Nested Relational Databases 13

Conversion of Databases **14**

Mapping Hierarchical Data to Nested Relational Databases 14

Performance of Converted Legacy Systems 15

Appendix A: Questions and Answers **16**

Nested Relational Databases and the Industry 16

Nested Relational Database Technology 16

Nested Relational Database Compatibility 17

Bibliographical References **18**

Nested Relational Databases

Introduction

Abstract

This paper discusses technical advantages represented by nested relational database technology. We focus on the removal of the requirement that relational databases conform to the first normal form (atomic attributes). Nested relational databases are technically called non-first-normal-form (NF2) databases, but are commonly referred to as MultiValue or extended relational databases.

We explain the differences between traditional relational databases and the Rocket MultiValue databases (principally the ability to nest tables and store complex data structures). Next, we outline the implications of this technology for the user of relational databases. We conclude by answering questions that are

Rocket

frequently asked about nested relational databases. An extensive bibliography of published material concerning nested relational databases is provided.

Basic Relational Database Technology

The well-known and widely-used relational database systems, including the Rocket MultiValue databases, Rocket UniData®, Rocket UniVerse™ and Rocket D3®, all conform to the basic relational model first proposed by Dr. E. F. Codd in 1969, and since refined by Codd and others. The relational model, unlike other data models, has a rigorous mathematical definition that is beyond the scope of this discussion. However, the basic technology shared by all relational databases can be summarized as:

- The database system maintains a clear distinction between the logical views of the data presented to the user and the physical structure of the data, as it is stored. The user does not need to understand the physical structure of the data in order to access and manage the data in the database.
- A simple logical data structure is easily understood by users who are not database technologists. Data are stored in tables. Each cell of a table contains an attribute value. Attributes in the same column are members of a set. Attributes in the same row are members of an ordered n-tuple. The n-tuples in the table form a relation, from which comes the term relational as applied to databases. Each table has one or more columns that contain the key to the table. The attributes in the key uniquely identify each relation.

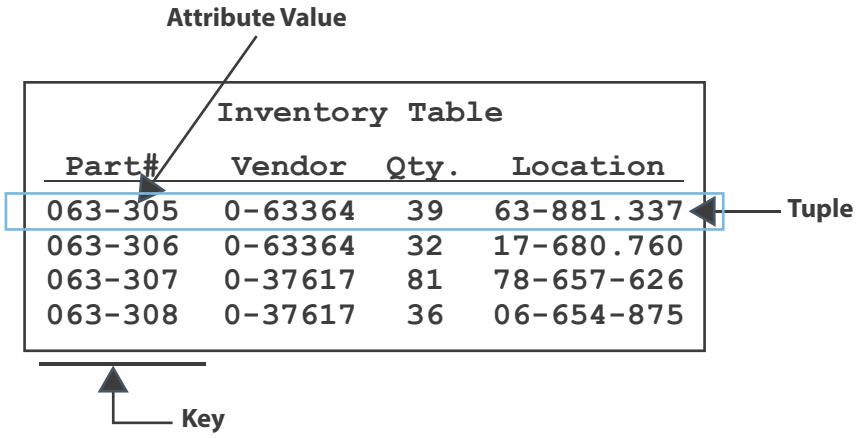

Figure 1. The relational model provides a simple, logical structure—the table—that stores attributes in rows and columns. The n-tuples of a row are a relation, from which the model gets its name.

- A high-level language is provided for accessing the sets (rows and columns) of the table, and for joining (combining) tables that have a common set of attributes (one or more columns containing the same attributes). The SQL language, which has been standardized by the American National Standards Institute (ANSI), fulfills this role, although many vendors provide other methods for accessing data.

Codd's proposal went considerably beyond just providing this model, however. It also included:

- Relational calculus, which is a mathematically rigorous definition of the "set operations" that a relational database should support for manipulation of tables.

  Rules defining how a relational database should operate. The rules cover matters ranging from the database access that must be provided for users to issues of data security.

All of the commercial relational databases, including the nested relational databases, conform to the relational calculus. A database that did not conform would produce results that were inconsistent with other relational databases. All of the commercial relational databases conform to a great majority of the rules (100% compliance is not necessary and no relational database attains it). A SQL statement operating on a given set of data produces consistent results when run on any of the commercial relational databases.

Normalization of Relational Databases

Normalization is a prominent aspect of relational database theory. Normalization addresses how data should be organized within a database in order to make the database as compact and easy to manage as possible, and to ensure that is produces consistent results. Normalization rules provide guidelines for defining the schema (design) of a relational database. Simply put, the rules specify how a database should be divided into tables and also how the tables should be linked together. The two major objectives of normalization are:

1. Minimize the duplication of data.
2. Minimize the number of attributes that must be updated when changes are made to the database, thereby making maintenance of the data easier and reducing the possibility of error.

Nested Table

Customer #	Customer Name	Order #	Part #	Qty.
AA2340987	Zedco, Inc.	93-1123	037617	81
			053135	36
		93-1154	063364	32
			087905	39
GV1203948	Alphabravo	93-2321	006776	72
			055622	81
			067587	29
MT1238979	Trisoar	93-2342	005449	33
			036893	52
			06525	29
			090643	33

Figure 2. This table includes nested data—the part numbers and quantities, also called repeating groups—and therefore does not conform to the first normal form (1NF).

Of the five ways to normalize data, called normal forms, three were initially defined by Codd[3,4,5] and two have since been defined by others. In order for a database to conform to the ***first normal form (1NF)***, attributes must be atomic; that is, an attribute must not be an n-tuple and therefore cannot be a set, list or, most importantly, a table or a complex data structure. This means that tables cannot be nested in a 1NF database. Figure 2 shows a nested table that does not conform to 1NF. Figure 3 illustrates how creating a separate customer and order table, and creating a relation in the customer table for each part that is ordered, eliminates the nesting of parts.

Adherence to the first normal form is a matter of the design of the database and the choices made during implementation. It is common to fully normalize a database during the design phase but then to denormalize during implementation in order to meet acceptable performance requirements. If the database does not support non-atomic attributes and chooses to denormalize for performance, the user must create multiple columns (e.g. order1, order2). This can make it very difficult to create simple, meaningful SQL statements to retrieve accurate results. In a nested relational database, the ability to support nested tables allows the user to create a single orders column, or a group of related columns (orders, parts, qty) that is nested or multi-valued. The nested relational SQL can then easily retrieve the correct results.

The ***second through fifth normal forms*** (hereafter the *higher normal forms*) define certain conditions for each of the normal forms that must be met in order for the database to conform to that normal form. For example, the second normal form declares that if (1) a table has a multivalued key and (2) contains an attribute that depends on only part of a multivalued key, then (3) that attribute should be moved to a separate table. The conversion of the tables in Figure 3 necessary to achieve 2NF conformance is shown

in Figure 4. The example in Figure 4 illustrates how conforming to 2NF can reduce the amount of data stored in the database and the number of attributes that must be modified when a change is made. In Figure 3, *customer number* is stored once for every part that is ordered. In Figure 4, *customer number* is stored only for each order. If it were necessary to change the number assigned to a customer, there would be many fewer fields that would require updating.

Customer Table

Customer #	Customer Name
AA2340987	Zedco, Inc.
GV1203948	Alphabravo
MT1238979	Trisoar

Order Table

Order #	Part #	Qty.	Customer #
93-1123	037617	81	AA2340987
93-1123	053135	36	AA2340987
93-1154	063364	32	AA2340987
93-1154	087905	39	AA2340987
93-2321	006776	72	GV1203948
93-2321	055622	81	GV1203948
93-2321	067587	29	GV1203948
93-2342	005449	33	MT1238979
93-2342	036893	52	MT1238979
93-2342	06525	29	MT1238979
93-4596	090643	33	MT1238979

Figure 3. To create a table that conforms to first normal form (1NF), the table of Figure 2 is divided into tables that contain no repeating groups.

Note, however, that if the characteristics of the data were such that many orders were for a single part, rather than for several parts as shown, the 2NF-conformant schema would actually require more storage because the *order number* key attribute must be stored twice for each order—once in the Order Table and again in the Order Customer Table. In that case, the schema of Figure 3 rather than of Figure 4 would be optimal.

Third, fourth and fifth normal form (3NF, 4NF and 5NF) similarly define increasingly stringent requirements, and adherence to each can reduce storage space, the number of updates required, or both. (For a detailed definition of the higher normal forms, see Codd[3,4,5.])

Customer Table

Customer #	Customer Name
AA2340987	Zedco, Inc.
GV1203948	Alphabravo
MT1238979	Trisoar

Order Customer

Order #	Customer #
93-1123	AA2340987
93-1154	AA2340987
93-2321	GV1203948
93-2342	MT1238979
93-4596	MT1238979

Order Table

Order #	Part #	Qty.
93-1123	037617	81
93-1123	053135	36
93-1154	063364	32
93-1154	087905	39
93-2321	006776	72
93-2321	055622	81
93-2321	067587	29
93-2342	005449	33
93-2342	036893	52
93-2342	06525	29
93-4596	090643	33

Figure 4. 2NF compliance is achieved by creating a separate Order Customer table to move customer number out of each Order relation.

While the physical conformance to the first normal form depends on the design of the database software, conformance to the higher normal forms depends on the database schema. With any relational database software, the designer can devise a schema that does not conform to any of the higher normal forms. It is impossible for the relational database software to determine with certainty that a given schema fails to conform; therefore, conformance is a concern of the database designer rather than a discipline that can be enforced by the database system. In any event, as illustrated, an apparent failure to conform does not necessarily mean that the database schema is less than optimal.

Relationship Relations

Figure 4 illustrates an artifice of normalization: the relationship relation. The Order Customer table contains relationship relations; so called because they do not serve to store data, but rather to keep track of relationships between tables (in this case, the relationship between customers and orders).

The requirement to have tables to store relationship relations has two disadvantages: (1) the additional tables confuse the design by adding elements that are not part of the fundamental data structure; and (2) the number of tables in the database increases. These factors cause maintenance and storage requirements to increase, cause system response times to lengthen because of the additional processing required, and make queries and other operations more difficult because SQL statements must be more complicated.

Limits of 1NF Relational Databases

With any relational database system, conformance to the higher normal forms is completely up to the database designer--the software imposes no constraints that prevent attaining an optimal schema, whether fully conformant or not. But databases that provide for the storage of atomic values give the designer no choice but to conform to 1NF. Conforming with the higher normal forms generally produces an optimal schema, albeit at the expense of greater complexity. However, database conformance with 1NF often increases the amount of storage used and makes maintenance more difficult. In addition, and most importantly, it greatly increases the processing required to produce results, while still making the schema more complex. Here's why (compare *Figure 2* and *Figure 4*):

1. The number of tables increases from one to three, including the table for a relationship relation.
2. Normalization of the tables requires the order number and customer number attributes to be stored twice for each order.
3. Producing a report to show the data (as in Figure 1) requires that the three tables be joined. Joins are highly compute-intensive operations.

For some potential users of relational databases, the joins that would be required to resolve relationship relations in 1NF databases would affect performance enough to preclude the use of relational databases. For example, 1NF relational databases are generally acknowledged to be unacceptable for CAD/CAM systems, which are used to design mechanical parts for manufacturing[11].

Why? One reason is that CAD/CAM data are inherently hierarchical in nature and the database structure used to store the part information must be traversed very quickly in order to display the part on the user's screen within an acceptable response time. Hundreds or thousands of join operations are required to display a complex part. These joins simply cannot be performed fast enough to provide acceptable display times. That is just one of several reasons 1NF databases are not used for CAD/CAM data. Such systems instead use proprietary hierarchical databases that provide high performance but are expensive to develop and maintain.

Apart from performance considerations, 1NF relational databases also have practical limitations for many applications. While any hierarchical or network database schema can validly be translated to a 1NF relational database schema, the practical considerations in doing so are daunting. Take for example a mechanical part. A hierarchical structure naturally and compactly stores the data that describe the part. The translation (mapping) of that hierarchical structure to a 1NF schema, however, is far from intuitive and leaves a confusing, awkward, complicated set of interrelated tables, including many tables for storing relationship relations. As a practical matter, such schemas are not possible to implement. These same considerations apply to many other types of data.

Eliminating the 1NF Constraint

Consideration of the limitations imposed by the 1NF constraint leads naturally to the question, "Can the 1NF constraint be removed from relational databases without invalidating the underlying relational model?"

As mentioned earlier, the relational database model has a mathematically rigorous definition to guarantee predictable, correct results on any database systems that faithfully implement the model. Detailed examination[1,7,9,10,12,13] has been made of the relational model with the 1NF constraint removed. The analysis has proven that the resulting model is equally robust. In other words, removing the 1NF constraint will not cause a relational database to produce invalid or inconsistent results as long as the database conforms to the higher normal forms.

When the 1NF constraint is removed—when nested tables are allowed—the database is technically described as a *non-first-normal form* (NF2) database but generally referred to as a nested relational or *extended relational* RDBMS. Database theoreticians accept nested or extended relational databases as valid relational databases. There is a body of literature on nested relational databases. Readers who wish to learn about the mathematical underpinnings of the nested relational model are encouraged to consult the Bibliographical References index.

Nested Relational Databases

Rocket Software is the industry leader with its implementation of the "best of breed" nested relational data model. A nested relational database management system is a relational database management system that has been extended in ways not possible for 1NF databases. SQL is supported as a data query language for nested relational databases. Since a nested relational database provides advanced capabilities, the SQL implementation must provide capabilities beyond those in the SQL standard, but with the standard being fully supported as a subset. The extensions support new relational operations that are not possible with a 1NF database.

It is important to note that nested relational databases can create 1NF views of a nested relational database to maintain semantic compatibility with applications and tools written for a 1NF database. Further, it is possible to optimize a nested relational database to provide the same performance for such applications as would be obtained if the applications were modified to use nested tables. The Rocket U2 nested relational databases implement those optimizations. That means it is possible to implement precisely the schema of any existing 1NF database as a nested relational database. Normally, the only reason to do so is to maintain compatibility to 1NF SQL. Database programmers and administrators must forego the benefits of the simpler schema of a nested relational database in that case.

Nested Tables

As mentioned in the Introduction, a basic tenet of the relational model is to provide the user a simple logical structure for data that is clearly distinct from whatever physical structure is employed. Nested relational databases fully support this tenet and further this goal by simplifying the logical structure of many databases by eliminating relationship relation tables.

Customer Table

Customer #	Customer Name
AA2340987	Zedco, Inc.
GV1203948	Alphabravo
MT1238979	Trisoar

Order Customer

Order #	Customer #
93-1123	AA2340987
93-1154	AA2340987
93-2321	GV1203948
93-2342	MT1238979
93-4596	MT1238979

Order Table

Order #	Part #	Qty.
93-1123	037617	81
93-1123	053135	36
93-1154	063364	32
93-1154	087905	39
93-2321	006776	72
93-2321	055622	81
93-2321	067587	29
93-2342	005449	33
93-2342	036893	52
93-2342	06525	29
93-4596	090643	33

Figure 5. The SQL statements required to define and access the attributes of these 1NF tables are given in the text.

Nested tables are implemented through straightforward and natural extensions to SQL. For example, the following SQL statement would create the 1NF tables illustrated in Figure 5:

```
CREATE TABLE CUSTOMER_TABLE (CUST# CHAR(9)
DISP ("Customer #"), CUST_NAME CHAR (40)
DISP ("Customer Name"));

CREATE TABLE ORDER_TABLE (ORDER_# NUMBER (6)
DISP ("ORDER #"), PART_# NUMBER (6) DISP ("Part
#"), QTY NUMBER (3) DISP("Qty."));

CREATE TABLE ORDER_CUST (CUST_# CHAR (9)
DISP ("Customer #"), ORDER_# DISP ("Order #")
Number (6));
```

By comparison, to define the nested table illustrated in Figure 6, the following SQL statement would replace the above:

```
CREATE TABLE NESTED_TABLE (CUST# CHAR (9) DISP ("Customer #),
CUST_NAME CHAR (40) DISP ("Customer Name"),
ORDER_# NUMBER (6) DISP ("Order #") SM ("MV") ASSOC ("ORDERS"),
PART_# NUMBER (6) DISP (Part #") SM ("MS") ASSOC ("ORDERS"),
QTY NUMBER (3) DISP ("Qty.") SM ("MS") ASSOC ("ORDERS"));
```

The Rocket MultiValue databases implement nested tables as repeating attributes and repeating groups of attributes that are associated. The SM clauses specify that the attribute is either repeating (*multivalued*—"MV") or a repeating group (*multi-subvalued*—"MS"). The ASSOC clause associates the attributes within a nested table. If desired, the Rocket MultiValue databases can support several nested tables within a base table.

Nested Table

Customer #	Customer Name	Order #	Part #	Qty.
AA2340987	Zedco, Inc.	93-1123	037617	81
			053135	36
		93-1154	063364	32
			087905	39
GV1203948	Alphabravo	93-2321	006776	72
			055622	81
			067587	29
MT1238979	Trisoar	93-2342	005449	33
			036893	52
			06525	29
			090643	33

Figure 6. The nested relational database table is equivalent to those in Figure 5. The SQL statements required to define and access this table and to convert it to 1NF are given in the text.

The following standard SQL statement would be required to process the 1NF tables of Figure 5 to produce the report shown in Figure 6:

```
SELECT CUSTOMER_TABLE.CUST#, CUST_NAME, ORDER_TABLE.ORDER_#, PART_#, QTY
FROM CUSTOMER_TABLE, ORDER_TABLE, ORDER_CUST
WHERE CUSTOMER_TABLE.CUST_# = ORDER_CUST.CUST_# AND ORDER_CUST.ORDER_# = ORDER
_TABLE.ORDER_#;
```

Several additional SQL statements also would be required to format the report as shown in Figure 6. Producing the same report from the nested table would require only the following SQL statement:

```
SELECT * FROM NESTED_TABLE;
```

Of course, Rocket can flatten a nested relational database (for example, when necessary to provide data to be stored in a 1NF database). To do so, the *UNNEST* clause of the SQL *SELECT* statement is used. The following SQL statement populates the Order Customer table shown in Figure 5.

```
INSERT INTO ORDER_CUSTOMER (CUST_#, ORDER_#)

SELECT CUST#, ORDER_# FROM NESTED_TABLE UNNEST ORDER_#);
```

Alternatively, the following SQL statement creates an ASCII text table to export the data of the Order Customer table to a 1NF database:

```
SELECT CUST#, ORDER_# FROM NESTED_TABLE UNNEST ORDER_# TO ASCII_TABLE;
```

Implicit Relationship Relations

As noted earlier, the requirement for tables (such as the Order Customer table in Figure 5) that store relationship relations is eliminated with nested relational databases. The information contained in relationship relations is required in nested relational databases just as it is in 1NF databases. In figure 6, it is still necessary to store the relationship between customers and orders; however, with a nested relational database, those relationship relations are *implicit,* rather than *explicit*.

Implicit relationship relations exist whenever one table is nested within another. Relationship relations are maintained automatically and very efficiently by the nested relational database. The designer, on the other hand, must define explicit relationship relations as additional tables in the database schema, with all of the additional complexity, maintenance and potential for error that the 1NF definition entails. Implicit relationship relations make the database schema simpler and allow it to directly reflect the relations between data in the database.

The requirement for explicit relationship relations does arise from time to time even when designing a nested relational database. In those cases the necessary tables can be defined, of course, just as with a 1NF database.

Technical Advantages of Nested Relational Databases

The following sections of this paper discuss the advantages of nested relational database technology for specific types of applications; however, nested relational databases have significant advantages for *any* database designed to store data that are naturally inclined to repeat:

- The database schema is simpler, with fewer tables to define and maintain.
- The database schema more directly mirrors the relationships inherent in the data stored, thereby making the schema easier to define and easier for others to understand.
- Many time-consuming and compute-intensive joins are eliminated.
- Redundant data for storing relationship relations are eliminated to save storage space.
- Accessing data is simpler and more straightforward. The required SQL statements are simpler and more direct.

Furthermore, these advantages come to nested relational database users with no offsetting disadvantages:

- A nested relational database is a true relational database. It is based on an equally valid mathematical model and provides all of the advantages of a 1NF database.
- 1NF-compliant databases can be implemented on a nested relational database and are fully supported.
- Existing SQL is fully supported because nested relational database SQL contains the ANSI-standard as a subset.

Conversions between nested and 1NF databases are easily accomplished.

Conversion of Databases

Virtually all enterprises that have information systems based on mainframe technology need to convert those systems to an open systems client/server environment or even to the Web paradigm. That includes many challenging tasks, and among the most difficult of those is the conversion of several large, mission-critical databases from a hierarchical database product such as IBM'S IMS or a networked database product.

IMS and IDMS have two distinct advantages compared to the most advanced 1NF relational databases:

1. The underlying relations in the enterprise's business data map easily to the database.
2. The databases are fast.

Nested relational databases ease the task of conversion to relational technology by providing the same advantages in a relational database. The following sections describe how.

Mapping Hierarchical Data to Nested Relational Databases

By their very nature, data relationships for business information tend to be hierarchical. Management structures are typically hierarchical, products are composed of hierarchies of parts and subassemblies, and markets are segmented hierarchically into geographical areas, and so on. The ease of mapping such structures onto a hierarchical database such as IMS is obvious—the designer simply maps a hierarchy to a hierarchical structure. The problems of mapping a hierarchy to the "flat" structure required for a 1NF relational database are equally obvious— because many relationship relation tables are required, as you have seen in the previous sections.

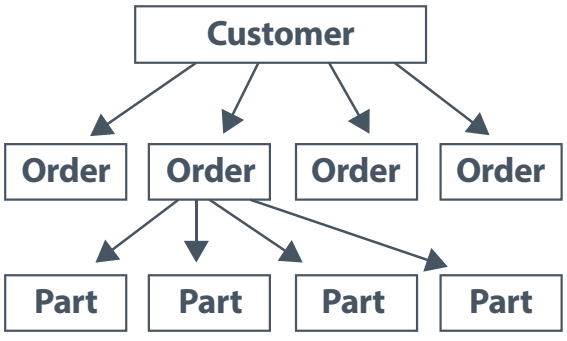

Figure 7. The previously used order database is inherently a hierarchical structure that maps naturally onto a nested relational database.

Rocket

The nesting of tables provided by a nested relational database allows for the direct mapping of hierarchically related information into a relational database. As illustrated in Figure 7, the customer order database is a hierarchical structure. It maps directly and easily into a nested relational database, but mapping into a 1NF database is much more complex and inefficient.

Performance of Converted Legacy Systems

The database products used for legacy databases may be proprietary, difficult and expensive to maintain, hard to change, and difficult to program, but they are also *fast*. A legacy database converted to a 1NF relational database often will not provide the responsiveness of the mainframe databases primarily because of the new requirement for performing joins. Joins are not required, for example, in hierarchical databases such as IMS. Relations between the different elements of the database are "hard-wired" into the hierarchical structure based on the results that the database is designed to produce, so the database produces those results very quickly. In any relational database, even a nested relational database, joins are slow, compute-intensive operations.

However, in a nested relational database, the need to perform joins is dramatically reduced by the database's ability to maintain implicit relationship relations, as explained earlier. In a sense, the implicit relationship relations are similar to the hard-wired hierarchical relations of IMS, in that they eliminate the need for joins and provide very fast data access. However, unlike the IMS hierarchy, the implicit relationship relations are as easily changed as anything else in a relational database. *Thus, nested relational databases combine the performance advantages of hierarchical systems with the flexibility and ease-of-use of relational databases.*

While nested relational databases cannot guarantee that relational database performance will equal that of proprietary mainframe systems, they do promise much better performance for legacy systems than can ever be obtained from a 1NF database.

extraneous tables used to store relationship relations. The extensions to standard SQL that support nested tables allow you to write simpler, shorter SQL statements.

Nested Relational Database Compatibility

Can I use my existing SQL with a nested relational database?

If the schema of the nested relational database matches the schema of the 1NF database for which the SQL was written, the SQL can be used without modification and will produce the same results. If you choose to take advantage of the advanced capabilities of the nested relational database to create nested relations, then you will need to modify only those portions of the existing SQL affected by those enhancements. In most cases, the SQL becomes simpler and more compact.

Can a nested relational database conform to the second, third and fourth normal forms? Does it need to?

A nested relational database can conform to all of the higher forms. The requirements for conformance are the same as those for a 1NF database.

Can you translate a nested relational database into a traditional unnested relational database?

First note that although a nested relational database need not conform to 1NF, there is nothing to prevent conformance. If necessary for compatibility, a nested relational database can store data with precisely the same schema used in a 1NF database from another vendor.

When it is necessary to convert a nested relational database containing nested relations to 1NF, the Rocket MultiValue database SQL and the data access language developed by research labs provide an "UNNEST" operation to easily convert a nested database to 1NF for or provide an unnested view of the nested relations.

Bibliographical References

1. Abiteboul, S., and Bidoit, N. "Non First Normal Form Relations to Represent Hierarchically Organized Data," Proceedings of 2nd ACMSIGACT/SIGMOD Symposium on Principles of Database Systems, March 1984, pp 191-200.
2. Arisawa, H., Moriya, K., and Miura, T. "Operations and the Properties On Non-First-Normal-Form Relational Databases." Proceedings of 9th International Conference On Very Large Data Bases. October 1983, pp 197-204.
3. Codd, E. "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM. Vol. 13, No. 6, June 1970, pp 377-387.
4. Codd, E. "Further Normalization of the Data Base Relational Model," Database Systems. Edited by Rustin, R. Prentice-Hall, pp 33-64.
5. Codd, E. "Relational Completeness of Data Base Sublanguages," Data Base Systems. Edited by Rustin, R. Prentice-Hall, pp 65-98.
6. Dadam, P., and Linnemann, V. "Advanced Information Management (AIM): Advanced Database Technology for Integrated Applications," IBM Systems Journal, Vol. 28 No. 4, 1989 pp 661-681.
7. Fischer, Patrick C. and Thomas, Stan J. "Operators for Non-First-Normal-Form Relations," Proceeding of IEEE COMPSAC '83, Chicago, November 1983, pp 464-475.
8. Garnett, L. and Tansel, A. "Equivalence of the Relational Algebra and Calculus for Nested Relations," Computers Math. Applic. Vol. 23, No. 10, 1992, pp 3-25.
9. Kitagawa, H., and Kunii, T.L. The Unnormalized Relational Data Model for Office Form Processor Design. Tokyo: Springer-Verlag, 1989.
10. Makinouchi, A. "A Consideration On Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model," Proceedings of 3rd International Conference on Very Large Data Bases, October 1977, pp 447-453.
11. Roth, M., Korth, H., and Silberschatz, A. "Extended Algebra and Calculus for Nested Relational Databases," ACM Transactions on Database Systems, Vol. 13 No. 4, December 1988, pp 389-417.
12. Schek, H., and Pistor, P. "Data Structures for an Integrated Data Base Management and Information Retrieval System," Proceedings of 8th International Conference on Very Large Data Bases. September 1982, pp 197-207.
13. Schek, H., and Scholl, M. "The Relational Model with Relation-Valued Attributes," Information Systems, Vol 11, No. 2. 1986, pp 137-147.

Appendix A: Questions and Answers

Nested Relational Databases and the Industry

Is it true that mainstream vendors are moving toward nested relational databases?

Because of the limitations of 1NF relational databases, especially for storing complex data structures, all commercial relational databases have begun adopting extended relational technology; however, Rocket has a technological lead of many years over its closest competitor.

Can't other database vendors just add nested relations onto their existing database engines?

Most relational database vendors have originally architected and implemented their products based on the assumption that attributes are always atomic. From their inception, the Rocket MultiValue databases were designed from the ground up to support nested relations. Adding this technology to existing traditional relational databases will necessitate either complete re-architecting of the database or emulation of this capability using the existing engine. Further, the Rocket MultiValue implementation, unlike others, is not limited to a single nested table.

Are nested relational databases accepted by the academic community as adhering to a valid relational model?

Yes. A large body of literature documents the validation of nested relational databases (see the Bibliographical References index.)

Nested Relational Database Technology

Are all nested relational databases relational?

Although the term "extended" or "nested" is usually applied only to relational databases, any database that stores non-atomic attributes is nested. For example, any CODASYL database is a nested database.

What is the difference between a nested relational database and a network database?

A network database such as a CODASYL database is not a relational database, although it may be termed a nested database. The differences between a true relational database and a non-relational database are: relational databases are based on a rigorous mathematical foundation and provide a standardized access language (SQL), other databases typically do not.

Isn't a nested relational database harder to use than a 1NF database?

Just the opposite is true. A nested relational database is easier to design, create, and use. The ability of a nested relational database to store nested relations means that the relations in the application data map much more directly onto the schema of the database. The ability to automatically and transparently maintain relationship relations reduces the overall complexity of the database by reducing the number of

white paper

Rocket

www.rocketsoftware.com
info@rocketsoftware.com