



## GREAT NEWS - THE RELATIONAL MODEL IS DEAD!

Michael Gorman, Whitmarsh Information Systems, Corp.

### 1. Introduction

Great news, the relational data model is dead! Well, not completely. It's that the relational data model, as we all know it through its linguistic expression, the SQL language, has been "dramatically extended" by the ANSI H2 Technical Committee on Database. The most interesting part of this "dramatic extension" into what is now called SQL/99, is that it has taken the SQL data model clearly into the past, and then beyond. To understand why these "dramatic extensions" take the SQL data model clearly into the past, this article presents a shortened version of a paper of the same name, Great News, The Relational Model is Dead, listed on the [Whitmarsh](#) website. This paper presents:

- An overview of data models,
- The four data models and the identification DBMSs,
- The SQL language,
- The SQL/99 language,
- SQL/99's impacts on the relational data model, and
- SQL/99's impacts of SQL/99 on database applications.

### 2. What Is a Data Model

A data model consists of three main components:

- Record Structures
- Relationships
- Operations

#### 2.1 Record Structures

A record structure (in relational terms, a table) represents a set of data fields (in relational terms, a column). There are several classes of data fields. These include:

- Single Value
- Multi-value
- Groups
- Repeating Groups
- Nested Repeating Groups

#### 2.2 Relationships

The second part of a data model is relationships. Relationships are explicit linguistic expressions that define the basis for interrelating records from different types of record structures. Relationships among values of multi-valued fields, instances of repeating groups, and instances of nested repeating groups within record structures are implicit and are completely controlled by the DBMS. The eight types of relationships that manage the relationships between records from different record structures are:

- One-to-one
- One-to-many (single member)
- One-to-many (multiple members)
- Many-to-many
- Singular, single member
- Singular, multiple member
- Inferential
- Recursive

### 2.3 Operations

The third part of a data model, operations consists of two types: Record structure operations and relationship operations. Record operations include the traditional ones of Add, Delete, and Modify. Relationship operations are of two classes and depend directly on whether the relationship mechanisms are DBMS defined and controlled, that is, member pointer arrays stored in the owner record, or are next, prior, and owner pointers stored in the member data records, or are shared value-based, that is, primary and foreign keys.

The choice of relationship implementation gives rise to two different styles of operations: record-at-a-time, or a multi-set of selected records. Record-at-a-time implies that the user's program navigates through the database and employs operations such as GET OWNER, GET NEXT, and GET MEMBER to accomplish the traversal. The set relationship operations are founded mainly on mathematical set operations upon selected sets of records. Included are PROJECT, DIVIDE, JOIN INTERSECTION and UNION. As DBMSs implemented these set processing operations they provides either explicit syntax or syntax support.

### 3. The Four Data Models

DBMSs from the various vendors have implemented different combinations of data model characteristics. From the late 1960s through the late 1980s, there was an attempt to standardize DBMSs that conformed to the network data model through the organization called CODASYL (Committee on Data Systems and Languages).

The documents produced by the CODASYL DDLC (data definition language committee) were called Journals of Development. They were not, however, ANSI standards. Thus, vendors adhered to these JODs "spiritually." Because there were neither ANSI standards nor conformance tests during this 1955-1970 time-frame to judge adherence to standards, every DBMS was somewhat different. Notwithstanding, four general data models arose.



**Figure 1. Characteristics of the Four Data Models**

[\*\*Click on the clipboard to see the diagram.\*\*](#)

Figure 1 presents the key characteristics of the four data models. This figure shows the four data models as the columns and the three main characteristics, that is, record structures, relationships, and operations as the rows of this table.



[\*\*Figure 2. Lexicon for the abbreviations in Figure 1.\*\*](#)  
[\*\*Click on the clipboard to see the diagram.\*\*](#)



[\*\*Figure 3. DBMSs by Data Model and Governing ANSI Standard\*\*](#)  
[\*\*Click on the clipboard to see the diagram.\*\*](#)

Figure 3 presents an enumeration of DBMSs by data models. In this figure, the dates relate to the earliest appearance of one or more of the DBMSs in a production status with one or more clients.

#### **4. The SQL Language**

The SQL language, originally known as the structured query language, was developed to support the relational data model. The language was created by IBM in the early 1970s. Ultimately, the language was made public domain by IBM and was then standardized by the ANSI NCITS (National Committee for Information Technology Standards) H2 Committee on Database in 1986. Note: H2 stands for nothing, it's just the committee's "primary key." The SQL language consists of the following main components:

- Database and data record structure definition including relationship integrity specification, and views.
- Data record operations for insert, update and delete, and relationship operations that accomplish JOIN, PROJECT, DIVIDE, INTERSECTION, and DIFFERENCE.
- Data record selection operations from a single data record or through nested subqueries to then select shared data value related data records.
- Privacy definition and control
- Concurrent update and retrieval data control

- Transaction processing

Because the SQL language is commonly employed through traditional programming languages like COBOL, SQL contains record-at-a-time processing commands, that is, cursor operations, that operate against selected sets of records.

The SQL/86 standard consists only of those facilities that the vendor-members of the H2 committee could agree upon. The goal was to standardize existing practice. From 1986 through to the next standard, 1989, other features were standardized including referential integrity. Referential integrity is an old concept and has been in CODASYL network systems since the late 1960s.

The next SQL standard was brought out in 1992. This was a major upgrade to the SQL language. The extensions were mainly in integrity constraints, multiple-language support, transaction processing, full referential integrity, and the like. The fundamental components of the underlying data structures and relationship processing remained the same. That is, tables that contain only single-valued fields.



**[Figure 4. Features of SQL/86, 89, and 92](#)**  
**[Click on the clipboard to see the diagram.](#)**

Figure 4 presents a table that shows the incremental features above the basic capabilities contained in SQL/86

## 5. SQL/99 Language

Starting in 1992, the H2 committee began the development of dramatic extensions to the SQL/92 standard. The greatest change in the standard is that it no longer adheres to the 1970 relational data model. The second biggest change is that the SQL/99 language now consists of individual parts that comprise a foundation and then a series of independently specified packages. The foundation components of the SQL/99 language include:

- Tables that have been enhanced to support new built-in data types (boolean, enumerated, extensions to character sets, translations, and collations)
- BLOB and CLOB data types
- Abstract Data Types (user defined data type with behavior, an encapsulated internal structure, and access characteristics of public, protected, or private)

- Array
- Row Types (table person (SSN, name(first, middle, last), address(street, city, state, zip(four, five))))))
- User Defined Functions
- Predicate extensions (for all, for some, similar to, cursor extensions, null values, assertions, view updatability, joins)
- Triggers
- Roles (enhancements to security), & Savepoints
- Recursion

The other parts of the SQL/99 language include:

- Call Level Interface—The SQL Call Level Interface is the set of language specifications used by DBMS vendors to enable direct SQL engine access through completely specified call routines. Microsoft, for example has implemented SQL/CLI and calls it ODBC.
- SQL/Multi Media (MM) Components— SQL/MM is itself a set of subparts that contain full specifications for a discrete set of data management functionality to address the data processing needs of full text, spatial, general purpose, and still image specifications and routines completely within the SQL language.
- SQL Persistent Stored Module Language Components--a complete embedded programming language to support the processing needs of its user defined data types, assertions, and triggers.
- SQL Transaction and Connection Management—this supports distributed processing, client-server, and of course the Internet, the ability to manage transactions is critical.

## 6. SQL/99's Impact on the Relational Data Model

As can be seen from the list of SQL/99 capabilities, it is no longer a simple language for defining, accessing and managing tables consisting only of single valued columns of data. With respect to the basic data model capabilities, the SQL/99 language more closely supports the independent logical file data model from the 1960s.

SQL/99 has, however, gone way beyond the capabilities of the independent logical file data model by incorporating facilities such as user-defined types, embedded programming language, and libraries of SQL/99 defined routines for areas like full text management and spatial data. To say that these SQL/99 extensions are mere "extended interpretations" of the relational data model is like saying that an intercontinental ballistic missile is merely an "extended interpretation" of

a spear.

SQL/99's impact on network and hierarchical data model DBMSs is significant. Network data model DBMSs have traditionally allowed complex data record structures with arrays, groups, repeating groups and nested repeating groups. A very unique characteristic of the SQL/99 data model is that it now allows arrays. In addition, the elements of the array are able to be outward references to other data. Since the order of the elements in an SQL/99 array is maintained by the SQL/99 DBMS, then the array, with its outward references, is essentially a CODASYL set. This is a dramatic departure from the relational data model.

The only remaining and viable network DBMSs are IDMS by Computer Associates and Oracle DBMS (formerly the VAX DBMS). Both have had an SQL language interface for about 10 years. How Computer Associates plans to take advantage of the existing IDMS facilities with SQL/99 is not known. A significant customer of Oracle's DBMS (formerly Vax DBMS from DEC) is Intel who uses the Consilium manufacturing package to manage computer chip manufacturing. How the Oracle Corporation plans to take advantage of the existing VAX DBMS facilities with SQL/99 is also not known.

The only two hierarchical DBMSs, System 2000 and IBM's IMS will likely not be impacted at all. System 2000 is no longer being advanced by SAS, and IBM has a full implementation of DB/2 on many different operating systems.

SQL/99's impact on independent logical file DBMSs, for example, Adabas, Focus, and Datacom/DB is significant. These DBMSs already support many of the SQL/99 data model facilities. It would seem that these DBMSs could rapidly conform to the new SQL/99 standard. If these vendors embrace the SQL/99 model, then these DBMSs could claim conformance sooner than the existing set of relational DBMSs.

Simply stated, the SQL/99 language defines a unique data model. It contains:

- The ability to model CODASYL sets,
- Many of the natural data clustering features of the hierarchical data model,
- Explicit many-to-many and inferential relationships like the independent logical file data model, and finally,
- The unique ability to directly model recursive relationships.

It therefore can only be said that the SQL/99 data model is unique unto itself. Clearly, it is not the relational data model, CODASYL network, hierarchical, or independent logical file data models. Simply, SQL/99 is a data model unto itself.

## **7. SQL/99's Impact on Database Applications**

For the past 20 years, database designers and implementors have

struggled with highly normalized databases that perform poorly. The only solution is to denormalize by collapsing hierarchies of non-redundant tables into a single flat table with replicated data. While these highly redundant collapsed tables speed data reporting, it slows updating, and also becomes a significant risk for data integrity. That is because the data is highly disbursed and is duplicated across these report-tuned denormalized database structures that are commonly known as data warehouses. For all these reasons, most organizations only allow reporting from data warehouse databases.

As DBMS vendors implement SQL/99, the database design process will transform itself from designing third normal table designs and then denormalizing these tables to enable cost effective reports to a set of database design activities similar to the ones that were commonly performed in database design efforts of the middle 1970s through the middle 1980s. There will have to be a greater knowledge of the application's processing to take advantage of the natural data structure hierarchies now possible within SQL/99 tables.

While processing speeds will dramatically improve with SQL/99 conforming DBMSs, the effort and processing time effort required to accomplish database redesigns and reorganizations will dramatically increase.

In short, we are returning to the past. That is, adopting the data structures of the network and independent logical file DBMSs. While we will see increased performance for well designed and highly tuned databases, we will also see the return of significant designer and analyst time for database design and redesigns.

Keith Hare of JCC Consulting ([www.jcc.com](http://www.jcc.com)), a long time member of H2 and a user of Vax DBMS products put it best when he said, "With SQL/99 you can get the best of both worlds and of course, you can get the worst of both worlds. It is up to the database practitioners to do the right thing."

*The long version of the paper is available from the "What's New" section of the [Whitemarsh website](#).*

---

*Michael M. Gorman, President of Whitemarsh Information Systems Corporation, has been involved in database and DBMS for almost 30 years. Mr. Gorman has been the Secretary of the ANSI Database Languages Committee, X3H2 for 19 years. X3H2 standardizes SQL. A full list of Whitemarsh's clients and products can be found on the web site, [www.wiscorp.com](http://www.wiscorp.com). The goal of the web site, WisWeb, is to make data management books, courses, methodologies, software, and metrics available to the database community through electronic publishing and downloading. WisWeb memberships are very reasonable and are designed for the individual, the ISD organization, universities/colleges, and professional training organizations.*

*Whitemarsh Information Systems Corporation  
2008 Althea Lane*

*Bowie, Maryland 20716-1518 USA*  
*Phone: +1.301.249.1142*  
*FAX: +1.301.249.8955*  
*Email: [mmgorman@wiscorp.com](mailto:mmgorman@wiscorp.com)*  
*WWWeb: <http://www.wiscorp.com>*

**[\[Return to the Article Archive\]](#)**

**The Data Administration Newsletter**  
**Robert S. Seiner - Publisher and Editor - [rseiner@tdan.com](mailto:rseiner@tdan.com)**