25 years of database history (starting in 1955)

6 December 2005

**A 1981 article in IBM's Journal of Research and Development gave me a much better perspective on how database systems got where they are.**

A 1981 article in IBM's Journal of Research and Development gave me a much better perspective on how database systems got where they are. The abstract of W.C. McGee's article [Data Base Technology](http://domino.research.ibm.com/tchjr/journalindex.nsf/0/18c2b2dadee8a44985256bfa0067f4d8?OpenDocument) tells us that "The evolution of data base technology over the past twenty-five years is surveyed, and major IBM contributions to this technology are identified and briefly noted." It put a lot of disjointed facts that I knew in perspective, showing how one thing led to another. All italicizing in indented block quotations below is his.

Around 1964, the term "data base" was "coined by workers in military information systems to denote collections of data shared by end-users of time sharing computer systems." In earlier days, each application had its own "master files" of data, so the concept of a data collection that could be shared by multiple applications was a new idea in efficiency.

The data structure classes of early systems were derived from punched card technology, and thus tended to be quite simple. A typical class was composed of *files* of *records* of a single type, with the record type being defined by an ordered set of fixed-length *fields*. Because of their regularity, such files are now referred to as *flat files*... Files were typically implemented on sequential storage media, such as magnetic tape.

Representation of one-to-many relationships was an early challenge.

The processing required to reflect such associations was not unlike punched card processing, involving many separate sorting and merging steps.

When punched cards were the only practical form of memory, the kinds of RAM-based interim data structures such as arrays and lists that we now create on the way to a final result all had to be done as separate piles of cards. While magnetic tape was an obvious step forward, separate runs for each sort operation and extraction must have still been pretty tedious.

Early structuring methods had the additional problem of being hardware-oriented. As a result, the languages used to operate on structures were similarly oriented.

He goes on to describe the evolution of the key "data structure classes" of databases: hierarchic (what we now call "hierarchical"), network, relational, and semantic. If you picture a database using each of these models as a collection of tables (or flat files), the great advantage of the relational model was the ability to create run-time connections between the tables—a JOIN. For hierarchic and network databases, they keys that represented links between tables had to be specified when you defined the tables. The advantage of the network model over the earlier hierarchic model was that the pattern of permanent joins did not need to fit into a tree structure. While I once worked at a [company that made a multi-platform hierarchic database](http://www.informationbuilders.com/), I never realized that hierarchic databases were around before computers had hard disks, when everything was done using tapes and punch cards. IBM began designing its IMS hierarchic database in 1966 for the Apollo space program, and it's [still around today](http://www-306.ibm.com/software/data/ims/).

Hierarchic databases were bad at storing many-to-many relationships, and in the mid-sixties the network model was developed. The use of the first commercial hard disks in computers enabled this more flexible access to data. While I've heard of the IBM IDMS product and GE's IDS that McGee mentions, I've never heard of "the TOTAL DBMS of CINCOM, perhaps the most widely used DBMS in the world today."

In the mid-1960s, a number of investigators began to grow dissatisfied with the hardware orientation of then extant data structuring methods, and in particular with the manner in which pointers and similar devices for implementing entity associations were being exposed to the users.

Mathematical approaches that applied set theory to data management used tables to represent sets of entities with attributes.

The key new concepts in the entity set method were the simplicity of the structures it provided and the use of entity identifiers (rather than pointers or hardware-dictated structures)... In the late 1960s, [IBM's] E.F. Codd noted that an entity set could be viewed as a mathematical relation on a set of domains D1, D2,. . .,D*n*, where each domain corresponds to a different property of the entity set.

This led to the relational database model.

Aside from the mathematical relation parallel, Codd's major contribution to data structures was the introduction of the notions of *normalization* and *normal forms*... To avoid update anomalies, Codd recommended that all information be represented in third normal form. While this conclusion may seem obvious today[1980!], it should be remembered that at the time the recommendation was made, the relationship between data structures and information was not well understood. Codd's work in effect paved the way for much of the work done on information modeling in the past ten years.

It also paved the way for a 1977 startup in Redwood Shores, California, called Software Development Laboratories, that could completely commit to the relational model, unlike IBM, who had many big customers using IMS and IDMS on IBM mainframes. When writing his paper three years later, McGee saw no reason to mention this little company, which would go on to become Oracle Corporation and play an obviously huge role in the use of relational databases.

Codd characterized his methodology as a *data model* and thereby provided a concise term for an important but previously unarticulated data base concept, namely, the *combination* of a class of data structures and the operations allowed on the structures of the class... The term "model" has been applied retroactively to early data structuring methods, so that, for example, we now speak of "hierarchic models" and "network models" as well as the relational model.

Many today consider the main choices of data models to be relational databases versus object-oriented models, with relational models having the performance edge because of the regular structure of the data. It's ironic to read McGee describe the performance problems of early relational databases; then, as now, higher levels of abstraction required more cycles—I guess those "hardware-dictated structures" had a payoff after all!

Speaking of object-oriented databases, note that McGee's snapshot of the state of the art in 1980 names "semantic data structures" as the next step after relational databases. He describes Peter Chen's Entity Relationship Model as an example of a semantic model. Academic papers and database (or rather, "data base") textbooks of the time are full of talk of the value of this next higher level of abstraction. Some could argue that the object-oriented approach was either competition to or an outgrowth of this work; I don't have the background to make a case for either side. For a bit more irony, it's kind of funny in this day of "semantic web" advocacy to read the big promises made in the name of "semantic data base systems" back then.

McGee's paper covers the development of other important DBMS concepts, often at IBM, such as the concept of the transaction (1976), views and authorization (1975), and report generators. This last development is interesting enough that I'll cover it in a separate essay.

At the end, after an introduction to the basic problem of distributed databases, McGee's conclusion tells us:

The solution of these problems promises to make the next twenty-five years of database technology as eventful and stimulating as the past twenty-five years have been.

I wonder if he considers new database developments from 1980 to 2005 to have been as stimulating as those of the preceding twenty-five years. I'd be surprised if he did. While the role of computers in our lives has obviously leapt ahead in that period, the progress in database technology, outside of performance issues and progress on distributed databases, can't compare to all the developments of those first twenty-five years. Advances that led to applications like Google are part of full-text search and information retrieval, a separate field with its own history going back to the early nineteen-sixties. I'll write about that when I finish [something else that I'm reading](http://www.amazon.com/gp/product/0262025388/).

**2 Comments**

By [mudrii](http://www.gentoo.ro/) on [December 7, 2005 12:12 AM](file:///G:\AACS3013%20DDA_2017-18s1\AACS3013%20DDA_Lecture_Lecturer\Chapter%202%20Database%20Concepts\Related%20Materials\25%20years%20of%20database%20history%20(starting%20in%201955)%20-%20bobdc.blog.htm#comment-25)

Thank you for article it was very interesting to read and look into the past.  
Thanks  
Regards

By [Ken North](http://www.sqlsummit.com/) on [August 3, 2007 12:58 PM](file:///G:\AACS3013%20DDA_2017-18s1\AACS3013%20DDA_Lecture_Lecturer\Chapter%202%20Database%20Concepts\Related%20Materials\25%20years%20of%20database%20history%20(starting%20in%201955)%20-%20bobdc.blog.htm#comment-1109)

W.C. McGee's *IBM Systems Journal* article provides an interesting history. There is another publication that provides quite a bit of detail. In March 1976, *ACM Computing Surveys* published a special issue about "Data-Base Management Systems". Several authors contributed six articles about the evolution of databases, about relational, CODASYL and hierarchical databases, and a comparison between relational and CODASYL (network) database technology. Don Chamberlin of IBM, co-inventor of SQL and XQuery, was the author of the article about relational DBMSs. The March 1976 articles are in the ACM digital archive at:  
<http://portal.acm.org/ft_gateway.cfm?id=984386&type=pdf>  
  
You mentioned "Mathematical approaches that applied set theory to data management" before going into an explanation of Codd's relational theory. This presentation about technology trends includes information about computing history, including the origins of database and the relational model:  
[Software and Database Technology Trends (slide presentation)](http://www.webservicessummit.com/Trends/TechTrends1/ComputingTrends_part1.html)  
  
You'll find it acknowledges the contribution of David L. Childs. Codd's seminal paper about the relational model followed a Childs' paper about set theoretic data structures. In fact, Codd cited the Childs paper in his paper.

We discussed this history in comp.database.theory in 2004. That thread noted Childs' work wasn't widely published at the time because it was government-funded research with a restricted audience.

The relational model evolved over time. For example, by 1990, Chris Date was describing the relational model in three parts ("Introduction to Database Systems"):  
  
- relational data structure  
- relational integrity rules  
- relational algebra.  
  
Childs' 1968 papers and Codd's 1970 paper discussed structure (independent sets, no fixed structure, access by name instead of by pointers) and operations (union, restriction, etc.). Childs' papers included benchmark times for doing set operations on an IBM 7090. Codd's 1970 paper introduced normal forms, and his subsequent papers introduced the integrity rules.

What's interesting is the University of Michigan connection. Codd, Bing Yao, and Michael Stonebreaker were graduates. Some of the work done at University of Michigan during that time (Childs' STDS, Ash and Sibley's TRAMP relational memory) was for the CONCOMP project. It was funded by the US government and the research was available only to "qualified requesters".