Process-oriented or Data-oriented Development?

Thoughts on the 40th Anniversary of Database Management Systems

by Breogán Gonda
bgv@artech.com.uy

Foreword

I would not like the reader to feel frustrated by reading this work: I have never had the possibility or the vocation to see life go by. I have always taken up strong professional commitments and, because of this, I do not mean to appear as an objective person in the eyes of people.

Databases: some history and some reflections

1963 was a fertile year for IT. It was the year of the so-called 3rd Generation computers! Manufacturers, for the first time, started thinking that ordinary companies could benefit from the use of IT, which, up to then, was restricted to huge projects, many of which had a strategic value and were linked to the Cold War.

One day in 1963, Honeywell launched its H200, the first computer of the so-called 3rd Generation. Other companies quickly replied with other "3rd Generation" computers, or announcements about them (in some cases, they just mentioned some of its characteristics and showed sketches of their cabinets, because they had been taken by surprise and their projects were still in a primitive stage): the leader, IBM had its "/360", General Electric its "Compatibles 400" and "Compatibles 600", RCA its "Spectra" line, Univac launched new hardware and its legendary operative system "Exec", etc.

What was the most important thing about the 3rd Generation? Judging from its repercussions, the most important elements were, on the one hand, the formalization of software concepts in general and of the operating system in particular as something independent from hardware and, on the other hand, a strong push for standard programming languages (Cobol, Fortran and Algol) that theoretically already existed in 1959, but were not used in the actual development of applications.

Charles Bachman, the first ideas on databases, the first Database Management System (DBMS)

At the time, something very important happened that was not noticed by most people: In a pretty quiet way, General Electric in the United States and Bull – General Electric in the rest of the world – released the IDS (Integrated Data Store) which became the first Database Management System on the world market.

The IDS followed the ideas of Charles Bachman [1], the great Database pioneer. At the time, databases were taken by most people in the IT community as an exaggerated sophistication, only destined to a few very complex applications.

Some of us, however, thought from the very beginning that databases were destined to support **all** computer applications, while the majority did not take our positions very seriously.

What has happened during these 40 years? How has the launching of the IDS affected system development?

Let us quickly review, freely, Charles Bachman's main ideas:

The Database must have all of the company data.

All systems will interact with the Database.

The Database Management System must enable storage, updating, deletion, and both quick and simple recovery of those data. In order to do so, it must also have mechanisms for access and integrity control/assurance.

What were the integrity mechanisms anticipated by Bachman? Those of "entity integrity" (Each entity will have an explicit identifier, there will not be two instances of an entity with the same value as the identifier's) and those of "referential integrity" (which, in the IDS implementation, we could very well

characterize as: "each child will have at least one parent", one "parent may have zero, one or multiple children" and as a corollary: "children" may also be "parents".

The IDS implementation used to be a free network that was basically supported through random access by primary key, whereas all of the other forms of access and the integrity control/assurance were implemented through pointer chains.

If we match (only from the qualitative point of view) the renderings released by Bachman in the IDS with those from current Database management systems, we may be misled to think that very little has changed in 40 years: that is not actually the case, although we have to admit that Charles Bachman's ideas, put into practice for the first time in 1963, not through a paper, but through something much more solid: facts (the IDS Database Management System on the market), continue to be fully valid.

The market response to the IDS: multiple Database management systems

In the years that followed, a set of relatively formal Database Management Systems came into existence. The most important known developments occurred inside IBM and were two, very different from each other: the BOMP and the IMS.

The BOMP (Bill Of Materials Processor) was a specific-purpose system based on a network that had a much lower expressive power than Bachman's: 2-level network (may be characterized as a free network to which we add the following restriction: "children" are not "parents".

The IMS (Integrated Management System), originated in a great project from the space race in which IBM was involved, whose structure was a forest of trees (may be characterized as: every "child" has a single "parent").

IBM ended up going for the IMS. The BOMP team felt very frustrated, which lead, at the beginning of the second half of the 60's, to inspiring the creation of Cincom Systems, one of the first independent software companies, that launched its Database Management System: TOTAL, readopting the basic ideas of the BOMP.

TOTAL was initially implemented for IBM mainframes, but later received simple and solid implementations for other platforms (including the small IBM /3 mini computer, the first one in which I used it in 1976). TOTAL rapidly became the leader in number of installations.

Expressive power of the first Database management systems and their easy/difficult reorganization

At this point, there was a system with a great expressive power (IDS), but whose truly blocked implementation posed a great problem: how to reorganize a network, basically supported through pointer chains? A good solution was never found to meet this need (whatever the procedure used, we can theoretically conclude that the times will be huge).

As a consequence, the first trends seeking "stable databases" arose stating that for a given company or organization, there is a "stable Database" that meets all of its requirements, taking all the time necessary in the preliminary studies in order to assure that "the Database will never have to be structurally reorganized" and, moreover, leaving empty fields in all the records so as to be able to add fields to them without having to turn to the feared reorganization.

40 years later, it is clear that such "stable databases" do not exist (at least in companies that are not dead). Is this, however, a truth that is generally admitted?: I doubt it; many people go on believing and working with the same concepts that were in place 40 years ago. Even nowadays, believe it or not, most system development methodologies are based on "stable databases".

What happened to the IMS? Trees are much easier to reorganize than free networks, but their expressive power is quite less than theirs (the basic premise is that "reality is hierarchical", but this is a false premise; the correct premise should be: "the visions of reality that humans can handle at ease are hierarchical.)" The lack of Database expressive power places the burden on the programs, leaving in the hands of programmers a great deal of the browsing through the Database as well as the integrity control/assurance, which is both expensive and dangerous. IBM tries to correct the problem by replacing the original independent forest of trees by a strange network of trees that aim at each other at all levels, in a very chaotic way.

The IMS, in spite of a certain commercial success obtained and although it was used in large databases (measured in number of records), never was a solution for true corporate databases.

The choice of a hierarchical scheme for the IMS was a mistake. The fact that it was IBM who made this mistake was significantly important and it became a great hindrance in the development of Database management systems for many years.

What about TOTAL? TOTAL was between both of them: its 2-level networks had much less expressive power than the IDS free networks, but much higher power than the IMS trees. Likewise, its reorganization was more complicated and required more time than the IMS, but far less time than what was required for reorganizing the IDS free networks. Moreover, its simple and non-pretentious implementation paved the way for its availability for different computers and operating systems. As a consequence, TOTAL ended up dominating the mid-size companies market and opened the way for many small businesses.

The response to the structural complexity: index based systems

The difficulties in reorganization and the search for greater flexibility in data retrieval originated Database management systems with very low expressive power (particularly without any capacity to control/assure the referential integrity). These systems, however, were very easy to reorganize and were reasonably flexible in terms of data retrieval; little by little, they became much more efficient in terms of access: systems based on files with multiple indexes (mainly the Datacom from the Applied Data Research and the Adabas from the Software AG and, to a certain extent, the VSAM from IBM – essentially, a file administration system).

This scheme implies attributing a set of functions to the programmer for integrity assurance purposes, which is inefficient in terms of costs (money & time) and dangerous: What happens when a programmer, in a given program, either forgets or misinterprets a rule?, Who knows what the rules that really rule our Database and our systems are?

Justice in the United States and the software industry

At the end of the 60's, the decisions made by the United States justice system that force IBM to quote and sell Software and Hardware separately end up significantly fostering the independent software industry.

Database management systems become the first battlefield for many companies, usually independent from the manufacturers; each of these companies adds its own casuistry.

The industry develops (anarchically) and the complexity of Database management systems grows day by day.

The search for simplification and "usability", the dream of giving the power to final users: Codd and the relational model

However, at the same time, inside IBM a simplifying trend arises: Edgar F. Codd [2] seems to wonder about things like "is reality so complex or is it that humans complicate things uselessly in order to represent it?"

Codd wants to make databases available – in all aspects – for everybody, by plucking them out from the super-specialist sphere. He seeks to simplify the database design and use problem, so he introduces his relational model on the following basis: simple data representation (tables with columns made up of uniform and atomic elements), criteria for spotting (and eliminating or – if not – controlling) redundancy (normalization), rules and operators for automatically manipulating data (relational algebra, relational calculus).

IBM molds part of Codd's ideas into the SQL language specification (Structured Query Language) – which it publishes and releases for public use – and which will later become the world standard (and remains so, even today, although with minor modifications).

Codd's contribution is huge. IBM's Santa Teresa Laboratory publishes and/or sends, generously and unselfishly, different papers to all the researchers who have shown an interest in the subject, which become

a basis for all those of us who have worked in the development of relational Database management systems.

Relational databases

The IT community adopts the idea of relational databases enthusiastically, snobbishly and with no pragmatism whatsoever.

Everyone is in favor of them; everyone implements some kind of software following the relational model.

No one is concerned about efficiency, and especially, no one faces the fact that access mechanisms (in general) and indexes (in particular) are essential for the efficiency of relational databases (They were so in 1970; they are so today; and they will remain the same in the foreseeable future). No one assures that deterministic procedures may be established for optimally designing and building these mechanisms and indexes, and most people are not sure that they may or should be used in a form that is transparent to the programmer and to the programs (It took time, but optimizers have done so for years, better each time.).

The SQL level is too low: the fact that users have to know from which tables to withdraw the data and how to link those tables is not reasonable, and has been an important problem until nowadays.

Let us analyze this in an example. If we have a database with the following tables:

Clients (Client, Name, Address)
Products (Product, Description, Price, Stock)
Invoices (Invoice, Date, Client)
InvoicesLines (Invoice, Product, Amount)

And we want to obtain the following tabulation:

Client	Name	Invoice	Date	Product	Description	Amount

The SQL command that we must build to achieve this is the following:

Select Clients.Client, Name, Invoices.Invoice, Date, Products.Product, Description, Amount
Where Clients.Client = Invoices.Client and
Invoices. Product = Products.Product

Nevertheless, just by adding the support of a good names nomenclature to the SQL (such as the URA: Universal RelationalAssumption) and providing it with a minimum intelligence, this command could be substituted by the following:

Select Client, Name, Invoice, Date, Product, Description, Amount

We will obtain the same result writing less. Yes, of course, but the main difference does not lie in the fact of writing more or less: the first one is a clear case of a command to be written by an IT expert who knows in which tables are the different attributes located and how to link these tables in a valid way.

The second one is a command that can be written by a user: he simply specifies what are the columns he wants in his list and the order in which he wants them. He says what he needs and knows and he does not need to ask anyone to help him.

It seems clear that the second syntax is the best one, since it enables much more users to use it naturally.

But the differences are not yet exhausted: most of the time, a high level specification has advantages regarding a low level one, independently of what is written!

The big difference is that if some of the columns referred to in the select changes from one table to another, the first statement becomes wrong while the second one remains valid.

No one is concerned about the referential integrity: Codd did not define it explicitly and "all those things that Codd did not say" have become exaggeratedly important.

The SQL should actually solve these last two problems: nothing has been done about the first one, and only a few years ago the referential integrity was given support.

The inefficiency of relational Database management system prototypes. Casuistry implementations continue

In view of the inefficiency shown by the SQL prototypes, we hear things such as: "when magnetic bubble memories exist, the problem will be solved" (Actually, many years after the proven failure of magnetic bubble memories, the argument was still used.); this means that: the problem is deliberately ignored by putting its discussion off for later without having the slightest idea of how to solve it and without making serious efforts to do so.

We are in the second half of the 70's. As I have said, everyone is in favor of the relational model (as something that "fits"), but in parallel more-casuistry-based Database management systems are implemented.

The commercial war between IBM, on one hand, and Honeywell, General Electric, RCA, Univac, etc., on the other hand, has come to an end. IBM has won by far.

Many of the more sophisticated applications were supported by the IDS, and their users begin to fear for the future development and support of General Electric computers and wish an IDS version for the IBM platform, which results in the creation of the IDMS, a product that is isomorphic to the IDS, implemented by an independent company for the IBM mainframe

A brief important reflection: data/process reorganization

Please note that, so far, I have referred with special emphasis to database reorganization problems and I have not talked about the inadequacy problems of the pre-existing programs when structural modifications take place in databases. Actually, the former problems were so serious, that they blocked the way to see the enormous importance of the latter.

Data or process-oriented development?

Together with the development of databases and, particularly, of Database management systems, there is a lot of work focused on the construction of system development methodologies. A dichotomy arises: "data-oriented development" or "process-oriented development"?

Let us remember that databases were only relevant for huge users:; average-size companies did not use them, and IT people, in general, saw them as a useless sophistication and a waste of time fostered by "theoreticians who had never seen an application."

This situation may have contributed for the balance between data orientation and process orientation to end up decisively favoring the latter. One may reasonably argue in favor of the process orientation and say that it is more general:; all the casuistry is embraced by it, everything can be done. At the same time, its level is much lower and the development and maintenance costs are much higher.

Who were the leaders in process-oriented development? Djikstra, by introducing structured programming and Edward Yourdon, Larry Constantine, Tom De Marco, Chris Gane, Trish Sarson and others, by introducing the structured project and making it popular throughout the world [3].

What can we say about data-oriented development? The leaders of this trend have been Jean-Dominique Warnier, Ken Orr and Michael Jackson [4].

Unlike process-oriented development, which is casuistry-based and seeks to support all the peculiarities of each application by manual programming , data-oriented development stems from other premises:

Data and programs always have structures – very often known in advance- such as the following:

the data seen by users;

the data that a program needs for processing a given operation;

the stored data;

a program, the program required for doing something;

Can't we think about rules and operator for working with those structures? Yes, we can. Actually, that would simplify the development and maintenance of systems a whole lot and would also improve quality: Warnier, Orr, and Jackson did that.

Some personal reflections about data-oriented development

I worked a lot – at the end of the 60's and the beginning of the 70's – with Warnier and Orr's methods: we continued programming manually, but there were clear criteria for identifying the data structures and user visions and deriving from them the programs structure and establishing their commands.

Anyway, my basic experience with this approach goes back to a time in which applications were fundamentally "batch", and used conventional files. If, instead of files, relational databases had been used, I believe that this approach would have clearly predominated. But you can't remake history!

I must confess that I did not have enough faith or dedication in using this approach – that I considered to be of a much higher level – when, suddenly, my clients' applications became fundamentally interactive and were supported by databases, and this is something for which I regret: at the time I decided to go – mistakenly – for process-oriented development.

At the same time, I wonder: why didn't Warnier, Orr, or Jackson – who seemed to have everything much clearer than the rest – take some additional steps in order to establish some things that were far more advanced (that I find obvious today, but which clearly are not – or, at least, they were not at the time)?: Database design stemming from the user visions, automatic program generation, etc. Actually, their methodologies "informally" led us to do all this manually and yet systematically, in a natural and simple way.

I would like to document all this, not as a complaint for what they did not do, but rather as a way to thank them for the invaluable introduction to a sort of rigorous representation of data structures, and, particularly, to the user visions of data and programs, which are essential elements for any data-oriented scheme (or, at a higher level, based on knowledge) and, particularly, for GeneXus [5].

In the 70's the dispute was won by those who held the process-orientation banner, which was the casuistry-based banner, the banner of the manual-programming and non-systematic orientation, which did away with all kinds of high-level operators.

But going back to databases: in 1979 ORACLE!

One day, in 1979, a small company whose original name I forget, and which later adopted its product's name, changed the world: it launched a relational Database Management System based on the SQL language that efficiently worked in a small computer. Oracle was the name of the product! Suddenly, the relational databases that everyone had been talking about became true: From then on, nothing would be the same (fortunately)!

The launching of Oracle had enormous repercussions: all of the others manufacturers had to put their feet on the ground and face that with the relational technology and **all the available means,** very important things could be done and they embarked in trying to do them.

During the 80's, after all manufacturers had claimed that their products (as they were) "were relational", a lot of work was done in implementing **truly** relational Database management systems. Potential users, however, did not know this and the actual use of Database management systems was very limited.

When we launched Genexus, in 1989, many people saw our taking of all of our clients' development to relational databases as an exaggerated sophistication that would force them to lose efficiency in running time. Some people told us: "we like Genexus, but we refuse to be forced to use a Database."

This situation changes decisively at the beginning of the 90's: suddenly, **all** systems start being developed on relational databases.

What has happened since 1990?

What has happened since 1990? What was (and will be) the impact of these events on the market, on the general trends, and on the possible elucidation of the old process-orientation vs. data-orientation discussion?

In the database area, the changes were very important but relatively not very visible, whereas in the process area, the object orientation was introduced, which has been very welcomed worldwide. However, and maybe most importantly during this time, there have been other things: the creation of "execution platforms" and the XML or, better yet, of a new orientation: "the message orientation."

Database management systems

Database management systems have evolved quite a lot. In what aspects? Mainly in terms of what is out of the users' sight. Nowadays they are much more solid; they are really solid! Their availability, security, efficiency and scalability have made a lot of progress, plus they work very well on very different software and hardware combinations, which are much cheaper than the traditional ones.

Concerning the functionality, there is a new, very important standard characteristic: the automatic definition and control of the referential integrity at the Database Management System level. And little more or nothing else that is really important.

All manufacturers support some sort of stored procedures, but in ways that are incompatible among each other.

As a consequence of this, the number of software houses focused on Database management systems has plummeted: only very big companies, with large investments, can offer products that are solid, efficient, and scalable enough. The main manufacturers and products nowadays are IBM (DB2 and Informix), Microsoft (SQL Server), and Oracle.

The elimination of small companies that manufacture Database management systems, together with the existence of a robust SQL standard, reduce the room for innovation more and more.

Sometimes leaders, instead of strengthening leadership in their vertical market through innovation, try to use the power that this position gives them for becoming stronger in other lateral markets such as execution platforms, application servers and, even application packages or development and outsourcing services. Most of the time, these policies isolate them from software development houses (their natural business partners) and end up acting against the development of new features in Database management systems. Everything becomes more conservative: great solidity and little innovation.

At this point in time an obvious question arises: will the current SQL standard be substantially modified, within a reasonable timeframe, so as to be provided with new high-level operators and other characteristics that will pave the way for an "intelligent" behavior of databases?

The general trend nowadays is supporting ordinary programming languages (Java, C#, etc.) for the writing of stored procedures. This means that in the future we may associate much more transportable logic to the Database, but always write it in low-level languages, which implies manual programming.

At the same time, two problems arise: the traditional security problems have been appropriately solved through good user identification and specification of what they are authorized to do with the data, but our Database management systems, for instance, are not immune to viruses. This is an area in which large manufacturers will have to work a whole lot.

Moreover, the spread of the Linux operative system has brought about the success of new very low-cost or free Database management systems, without the solidity and scalability aspirations of the great leaders. Postgres and MySQL are worth highlighting among these systems.

But today there is no more discussion as regards whether to use databases or not: all applications throughout the world are developed on databases; efficiency and scalability are incomparably higher

if we look at what was obtained with conventional files, and last, there are clear leaders that dominate the market.

The forms of use have changed over time; this has been influenced in small part by the available technologies and the needs of users, and largely influenced by the preferences and trends of IT professionals, from the original centralized framework, going through the Client/Server framework, and following, nowadays, with multi-server frameworks oriented to the network and Web Services, which have adopted the shape of execution platforms that will dominate the IT world in the foreseeable future: Java and .net:

In the mid 80's, when all applications worked in centralized architectures, the new "client/server architecture" concept arose.

It was about rationalizing and dividing data processing between the central server and the microcomputers (clients) and represented important advances, in spite of the errors that all of us made at the first stages of its implementation.

The firm that introduced and developed the concept was Sybase. But the market did not adopted this architecture until 1995, when Sybase was no longer in its best moment to fight with the strong competence.

In 1996, with the support of multiple software companies such as IBM and Oracle, Sun. launched the Java language and the Java development and execution platform.

In 2001, Microsoft launched the C# language and the .net development and execution platform, a sophisticated implementation of Java's ideas, with similar purposes and qualitative results.

In 2001 IBM, bought Informix. This, together with its leadership with its OS/390 and OS/400 systems, made it the major Database Management Systems manufacturer.

In 2002, Microsoft launched a new standard for the communications between applications and database for its .net platform (DataSets and DataAdapters). It made programming easier and means a great help for the writing of highly scalable applications.

Process orientation

Process orientation, which was once sustained by structural methodologies, has given way to object-oriented development.

Object-oriented development, supported by the existence of object-oriented programming languages, has clearly imposed itself and brought about huge advantages in programming, concerning in-memory data. Under these conditions, the advantages over traditional processes are huge.

There is a link that is still weak in object-orientation: How to obtain a good communication with the Database? There was quite a lot of experimenting, for some time, with the so-called object-oriented Database management systems (OODBMS),, but the results obtained were just not adequate.

Nowadays, it is clear for everyone that relational Database management systems dominate the data world and will continue doing so in the foreseeable future. A lot of work is being done in the creation of mechanisms that will enable object-oriented programs and relational databases (ORM: Object Relational Mapping) to live together. Although there are isolated achievements, we are far from obtaining a standard.

Execution Platforms

40 years ago there were multiple operative systems. Each manufacturer had their own. In turn, each one of them had to be capable of directly seeking to meet all of their customers' needs.

When the software industry originated, this situation started changing: clients had their needs met by a group of suppliers. However, an operative system would have only been attractive for the independent software industry if it had had many users.

In the mid-70's, many manufacturers were forced by the facts to abandon their operative systems and adopt other. Only those manufacturers that had very large databases installed (such as IBM OS/390, IBM

OS/400) survived, and some relatively independent from hardware manufacturers strongly grew: Windows, Unix , and lately Linux.

Each operative system has its own complexities. One more step is thinking about execution platforms that would work over them and would be much more user-friendly.

In 1996, Sun launches its (object-oriented) Java language and its Java development and execution platform. As a language, despite its important characteristics, it is just one more. But as an execution platform, it implies something totally new: it is the first time that a friendly environment is provided for developing and executing applications with total independence from the hardware. Although the concept was rapidly and generally adopted, the development of real Java applications took longer than expected. Today it is a reality.

In 2001, Microsoft launches .net, its new development and execution platform, with its own object-oriented languages C# and Visual Studio, and a number of third-party languages.

Why would the client use one of these platforms? For a number of reasons (that, in general, exceed the purpose of this work), the most important one being that: they provide automatic updating and installation of applications in "clients" workstations at truly much lower operating costs.

I believe that in the coming years, the market will be the shared between these two platforms, and that the operative systems will turn into commodities. At the same time, the adoption of these platforms reinforces the use of object-oriented languages.

XML and "message orientation"

In 1999, another player comes into the picture: XML, a system for defining and manipulating self-described messages. Another very important trend arises: "message orientation".

From the moment it was announced, XML became rapidly adopted by all of the manufacturers, which turned it into a standard, first informally and then formally. Its use has spread at a speed greater than the traditional speed of innovations: there really was a huge unmet need for a standard message system!

XML has a wide variety of uses; most of them take place out of the users' sight. However, it sometimes becomes the main player, for instance, in Web Services. Moreover, it will have very important derivations in the construction of truly extended databases: my Database, that of my suppliers/clients, that I access through XML and a set of Web Services of general interest that I "use" under the form of XML messages.

But if we go a little deeper into this, we will see that XML can become an important player in the old process-orientation vs. data-orientation discussion: many complicated processes, in the end, aim at providing us with a datum. Can't we think of this particular datum as provided by an XML message (data provider) that we will be able to specify in a form that is 100% declarative? I think so.

Once again, however, the "hierarchical databases" arise: manufacturers of Database management systems are announcing expansions of the SQL that will enable us to store and manipulate XML messages. I hope that this characteristic will be used with due prudence (that it will be circumscribed to very specific cases such as textual document support).

Summary:

Relational Database management systems have become more solid in the last few years; nowadays, talking about another type of Database Management System is just unthinkable.

Robust standards have been configured that, when strictly used, enable application portability.

Modifying those standards substantially is not an easy task. In particular, it is very unlikely that we will have much more qualitatively evolved databases in the foreseeable future.

However, we are very likely to obtain much more soundness: availability, security, efficiency, scalability and resource optimization, and to neutralize new threats, such as those posed by innovative viruses.

In parallel, object-oriented programming has become more generalized (in particular, strongly boosted by the predicted success of the Java and .net platforms) and it is vital that the problem of the natural connection between the programs and the Database (Object Relational Mapping) be rapidly solved.

Message orientation might be the missing component. We may not have a clearly process-oriented future nor a data-oriented one, but rather a piece of reality that behaves as a set of messages.

Today's Database management systems and Bachman and Codd's ideas

With all this, however, have we reached our destination? Is our aspiration – nowadays – the same as Charles Bachman's in 1963?: if so, if we are going to access databases through programs where everything is mostly done manually, then everything has been sorted out (and very well sorted out, because we have good levels of efficiency, availability, security, scalability, accident recovery, etc., and moreover, we also have reasonable means to question the Database). Bachman's dream have come true! But this took almost 40 years !

If, nonetheless, we think like Edgar F. Codd in 1970, and we wish to spread the use of databases throughout the world, then we are very far away. The so-widely-proclaimed and widespread "user age" seems to be very distant still!

My idea of the future

I firmly believe in a paradigm that is different form the current one, from the one represented by the current "State of the Art".

I believe in "intelligent databases" that will allow us to declaratively feed all the required knowledge (mainly, "business rules", "precedence and/or flow rules" and "authorization rules" with all the breadth embraced by those concepts) so that any user, without any technical knowledge whatsoever, will be able to do everything they want and is authorized to do, in a simple way and at any time, without having to program.

You may say that what I am saying is very difficult. I may agree. But let us refine this response a bit more: Is it very difficult because we lack the necessary technology? I do not think so: it is very difficult because it does not fit into the current State of the Art, and because of the general trend in a market that seems to be very conservative.

Sometimes, changing paradigms is much harder than developing the required technology for the new paradigm to come true.

Only some of the big players in the field of Database management systems (IBM, Microsoft, Oracle, any other?), are able to successfully drop the SQL standard and go for solutions that are much more evolved. But this is a great risk for a leader, who is almost always led to lean on the conservative side by success. Which of them will dare?

If some of these leaders takes the risk, we will have "intelligent databases" very soon.

Otherwise the current SQL standard will live a long and relatively quiet life.In this case, the problem will not be solved with it, but with rather knowledge-based, higher-level systems, automatically "operating" on the SQL.

I firmly believe that the replacement of the current paradigm (oriented to programmers and to manual programming for 40 years) by a knowledge-base paradigm is something inescapable. I believe that this replacement will be inescapable because manual programming is becoming less feasible as time goes by and it implies low levels of development productivity as well as dramatic system maintenance costs (time and money). Many times, it forces companies to sacrifice their individuality and give up competitive advantages in order to adopt standard products with consequent rigidity, because it is impossible for them to build and maintain the products that they really need.

Some of the most developed countries try to solve the problem by importing labor so that they can reduce their costs.

In parallel, in order to lower their great costs, many companies have decided to transfer their programming, and, little by little, their development (which in some cases results in total outsourcing) to countries where there is very cheap, relatively skilled labor. More and more countries are struggling for this market.

This trend causes final users to, little by little, lose freedom and is a big threat for the software industry in most of the more developed countries, and also in Latin America.

However, these attempts to lower costs are just palliatives: the real solution is the insertion of a kind of technology that will make well-remunerated professionals to be truly efficient at world levels without having to leave their countries.

How long will it take us to have intelligent databases? How long will it take us to truly get to the long-time proclaimed "user age"?

I do not know, but **it is not a matter of technology: the technology is there. It is a matter of attitude.**

From our company, we have taken the first steps with GeneXus and nowadays there are thousands of large applications worldwide, mainly critical mission systems, which are automatically developed and maintained based on knowledge; not a single line of code in low-level languages has been written. There is a lot more to do, but the feasibility of the new program has been shown. That is not a small accomplishment!

Bibliography

I would like to clarify some things about the bibliography used for this work:

It is difficult to provide a concrete bibliography when not dealing with very specific topics, but rather with a vast life experience of more than 40 years in the profession, focused on unconventional things from the very beginning.

This task becomes even more complex since the author, because of his professional activity, has had the privilege of participating in seminars and informal idea exchanges with many of the main figures that shaped the history of databases and system development methodologies, and/or has had access to documents that some of these people did not publish (in some cases, years before the authors finally published them).

There is, however, another growing problem as we speak: 40 years ago, there was a reasonable balance between the research done by manufacturers and the research done by universities around the world. Little by little, this situation has changed, and nowadays, the relative participation of universities in IT research has diminished quite a lot.

This situation is not good; nevertheless, it is the world reality and its first negative consequence is very important: a shortage of timely publications.

The industry is interested in doing innovative things, in outdoing their competitors, and very less often, in timely publishing their technological discoveries. Nowadays, in order to be updated on the technological edge, we have to rely on our own research and our good relations with other companies and other researchers, since generally speaking, when something is published, it is already too late for making the appropriate decisions.

Taking all of this into consideration, I find that only listing a bibliography is something completely inefficient; moreover, nowadays, many of the uncertainties that reigned at the time have been dissipated; readers might find this kind of listing quite useless. This is the reason why, apart from referring to a few documents and books that continue being of interest in the present day, I would like to refer today, as an explanation and as a token of gratitude, to events and people that have contributed with their perspectives.

Published books and papers:

[1] Charles Bachman

The Integrated Data Store, a General Purpose Programming System for Random Access Memories, General Electric, 1964.

Integrated Data Store Application Manual, General Electric, 1966.

[2] Edgar F. Codd

Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks, IBM Research Report, Santa Teresa Lab, San José, California, RJ599, 1969.

A Relational Model of Data for Large Shared Data Banks, CACM 13:6 1970.

Further Normalization of the Data Base Relational Model, IBM Research Report, Santa Teresa Lab, San José, California, RJ909, 1971.
The Relational Model for Database Management, Addison-Wesley, 1990.

[3] Process-oriented development

Edward Yourdon
A Case Study in Structured Programming – Redesign of a Payroll System, Proceedings of the IEEE Compcon conference, 1975, New York, IEEE 1975.
Techniques of Program Structure and Design, Englewood Cliffs, N.J.: Prentice-Hall, 1975.
Managing the Structured Techniques, New York, Yourdon Press, 1979.
Edward Yourdon & Larry Constantine:
Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design, New York, Yourdon Press, 1978.
Tom De Marco
Structured Analysis and System Specification, New York, Yourdon Press, 1979.
Chris Gane & Trish Sarson:
Structured Systems Analysis: Tool and Techniques, New York, Improved Systems Technologies, 1977.

[4] Data-oriented development

Jean-Dominique Warnier
Les procedures de traitement et leurs donnes, Les Editions D´Organization, Paris, 1975.
La transformation des programes, Les Editions D´Organization, Paris, 1976.
The Logical Construction of Programs, New York, van Nostrand Reinhold, 1974.
Entrainement a la programmation, Les Editions D´Organization, Paris, 1979.
Ken Orr
Structured Systems Development, New York, Yourdon Press, 1977.
Michael Jackson
Principles of Program Design, New York, Academy Press, 1975.
System Development, Prentice-Hall International, 1983.

[5] Knowledge Administration

ARTech: Genexus General View, 1989- 2003, www.genexus.com/documents/generalview.pdf
Breogán Gonda, Juan Nicolás Jodal
Desarrollo Incremental, 1989-2003, ARTech
Algunas reflexiones sobre los Modelos de Datos a los 35 años de su introducción, 1997, 1er Congreso Uruguayo de Informática, www.genexus.com/whitepapers/
Genexus: Philosophy, 2002-2003, Artech, www.genexus.com/whitepapers/

[6] Other:

C.J. Date: An Introduction to Database Systems, Addisson-Wesley, 1976.
David Maier: The Theory of Relational Databases,1983 Computer Science Press, Pitman, Publishing, 1983.
Jeffrey D. Ullman:Principles of Database and Knowledge-Base Systems, Computer Science Press, 1988.
Hermann E. Dolder: Diseño de Databases utilizando conceptos y técnicas de Inteligencia Artificial, EUDEBA, 1987.
Morton M. Astrahan, Mike W. Blasgen, Donald D. Chamberlin, Kapali P. Eswaran, Jim Gray, Patricia P. Griffiths, W. Frank King III, Raymond A. Lorie, Paul R. McJones, James W. Mehl, Gianfranco R. Putzolu, Irving L. Traiger, Bradford W. Wade, Vera Watson: System R: Relational Approach to Database Management. TODS 1(2): 97-137 (1976).
Gerald Held, Michael Stonebraker, Eugene Wong: INGRES: A Relational Data Base Management System. AFIPS NCC 1975: 409-416.
Bo Sundgren: Theory of Databases, Petrocelli/Charter, New York, 1975.
Borge Langefords, Bo Sundgren: Information Systems Architecture, Mason/Charter, New York, 1974.
Donald E. Knuth: The Art of Computer Programming, Addison-Wesley, 1968-1973.

Unpublished documents.

Several from Jean Dominique Warnier.
Several from Charles Bachman.
Many works developed by the IBM's Santa Teresa Laboratory.
Many works from other research laboratories.

Courses, seminars and personal discussions.

Advanced Database Seminar held at Pontificia Universidad Católica of Rio de Janeiro in 1976 with the participation of Codd, Date, Blasgen, Stonebraker, Furtado, etc.

Many seminars organized by SCI in Brazil, and as Technology Director of the organizing company, technical discussions with other lecturers such as Yourdon, Gane, De Marco and Date.

Personal discussions with my friend Hermann Dolder.

Personal discussions with my friend Ken Orr.

And, most especially, personal discussions with Nicolás Jodal and all of ARTech's team.