RDBMS Workshop: Technology Through 1983

Moderator:
Burton Grad

Recorded: June 12, 2007
Mountain View, California

CHM Reference number: X4069.2007

© 2007 Computer History Museum

Table of Contents

RDBMS Workshop: Technology through 1983

Conducted by Software Industry SIG – Oral History Project

Abstract: A group of relational database technology pioneers from IBM, Oracle and Ingres discuss the origins and implementation of the relational model into deliverable products. The period covered was primarily from the early 1970s through 1983. They discuss the major technological problems which they had to deal with including optimization, concurrency control and query performance. There is a lengthy discussion of the various RDBMS projects in IBM and why the release of IBM's products was delayed and opened the door for Oracle and Ingres to enter the market before IBM.

Participants:

Name	Affiliation
Burt Grad	Moderator
Greg Batti	Ingres
Michael Blasgen	IBM
Marilyn Bohl	IBM, Ingres
Paul Butterworth	Ingres
Don Chamberlin	IBM
Sharon Codd	IBM
Chris Date	IBM
Don Haderle	IBM
Roy Harrington	Informix
Jerry Held	Ingres, Tandem, Oracle
Ken Jacobs	Oracle
Bruce Lindsay	IBM
Jim Strickland	IBM
Moshe Zloof	IBM
Peter Capek	Historian, IBM Research
Michael Mahoney	Historian, Princeton University
Jan Phillips	SI SIG, DEC

 © 2007 Computer History Museum

Burton Grad: We're now opening the first technical session. We're going to cover up to about 1983 or so. I want to explore the technical developments that occurred from the initial paper by Codd. And we've heard about some of the influences and effects it had and how it started various other things. And since we didn't go deeply into it before, I want to make sure we do talk about the work that was going on in System R and how IBM's developments took place, talk about the implementations, talk about the new ideas and concepts that made the systems practical and workable, whether they were to the query languages, whether they were to the implementation process, or any other of the technical issues that you are aware of, and we'll get a chance to cover these in the next session.

Efficient Queries of the Database

Bruce Lindsay: Well, I think an important historical moment that's perhaps not been emphasized enough is that the relational model doesn't tell you how to do the query. And to me a very seminal paper was the paper by Mike Blasgen and Kapali Eswaren, who did a performance study of different join methods and actually showed that sometimes a nested-loop join was best, sometimes merge join was best. [The hash join hadn't been invented yet] To me it really said, "Golly, any implementation is going to have to make a choice."

Grad: In what sense, Bruce?

Lindsay: Given a query, there are several choices of more or less efficient algorithms for evaluating, say, a join predicate. Without the Blasgen/Eswaren study, I think you might have said, "Well, what choices do we have? We'll just do a nested loop. We'll get all the answers, right? End of story." Because it would give you the right answer. But this paper showed that indeed for some queries, merge join was the more efficient mechanism to evaluate the join predicate.

Grad: When was this paper published, Mike? Do you remember?

Michael Blasgen: The work was done at the end of 1974 and the beginning of 1975. It was published as a research report in its full-blown form looking about at as many as 15 different algorithms that would achieve the correct result. The paper that was actually published more widely in the IBM Systems Journal in 1976 only had about six or eight of the good methods.

Grad: About 1976 then?

Blasgen: That's when it was published. The work was done in 1974 and early 1975.

Grad: Well, maybe this is a way to start. What were some of the primary

implementation issues that were faced in whatever companies that were working on them?

System R Background

Blasgen: I'd like to set the scene that led to System R. In 1973, the people in IBM San Jose Research built the Sequel Phase Zero prototype. It was an implementation of SQL that supported ad hoc query and the full language as it had then been defined, even including features that were sort of on the edge like triggers. But that was to, I think, get everybody experienced with writing code and building systems. And then at the end of 1974 we started System R and that system had a property--really high performance--that was shared with the first implementation of FORTRAN in 1956 or 1957. I've always been interested in the history of FORTRAN. In fact, John Backus had his office was just a couple doors down from all of us working on System R. The 1957 FORTRAN compiler generated code that was just phenomenal. They believed that their competition was people writing assembly. They weren't going to let assembly-language programmers generate better code than they did. They made about 50 technological breakthroughs in developing that. The FORTRAN team also biased the language. For example, if you remember, the old FORTRAN is "IF (expression) 1, 2, 3." The expression would go to statement 1 if the expression was negative, 2 for zero, 3 for positive. Why is that? Because the IBM 704 had that as a single instruction, that is, a three-way branch was in the hardware. The FORTRAN team wasn't going to let you gain a one instruction advantage. So they actually defined the language in order to get maximum performance. To some extent, System R did all the same things. For example, one characteristic is System R generated machine code that was then executed. Even in its ad hoc query mode, we generated machine code and ran it. There was no interpreter, if you will. It was all compiled. Later, as we made the prototype a product, we actually changed that. I mean, the people in IBM Santa Teresa [now Silicon Valley Lab] changed it because they certainly didn't like the idea of actual bit strings of executable code that you pointed the instruction counter at, which is what--

Don Haderle: It was non-serviceable and it had lousy performance. Other than that it was not bad.

<laughter>

Blasgen: Don said that he didn't like aspects of our design at the time and we were so upset, but he was absolutely right. We were upset. Our code, we thought, was perfect.

Grad: That's an interesting story.

Blasgen: We couldn't stand to be criticized. He called our baby ugly, and I didn't like that. <laughs>

Haderle: Generating machine code created support issues. And they attacked the wrong area for performance.

Blasgen: I agree with you. But I'm just making a general point, which is sometimes these early systems bent over backwards to be competitive with, for example, the navigational alternatives because with the navigational guys, there's no decision-making. The code that supports the navigation just lets the navigation occur. In SQL, as Bruce just pointed out, there's a million ways to do it, so which one do you pick? And that was a whole big deal of this, and we spent a lot of effort trying to pick a good one.

Haderle: System R was a prototype that did exactly when a prototype is supposed to do, for example it revealed where the performance problems were. SQL is a dynamic language between the application and the database, and every time that you linked up between the two, you had to go and explore all of the variables over here and see if they matched the type that was in the database. It was all dynamic in that exchange. For every fetch, they could go and change something on this side. Every time that you linked up you had to go and check to see if the data type here matched the data type there. You had to do some conversion. There wasn't a strict binding over here that lasted long. System R showed where you spent gobs of time. While System R had spent effort in compilation to machine code, the total performance problem was more complicated. We redesigned the optimizer to generate plans that were executed be a low level interpreter instead of creating direct machine code. The small increase in path length was more than paid for by easier support and simpler design.

Grad: Now these are problems in efficiently answering the queries. Are these problems in how to organize the implementation of the code?

Haderle: No. System R came along as did others, and they put together a prototype and then we tried to commercialize the prototype. And then commercializing the prototype, because they were there first, you got the benefits of seeing what was not working. The Research team explored the fact that they could compile down to get it to run very fast in the database, but that was not the main performance obstacle for most applications. System R had a system for tracking application and database changes to know if a reoptimization and recompliation was required. We replaced that with a different system that was less complex and led to fewer recompiles. Finally we had to deal with new releases of the DBMS, and issues the prototype had not dealt with

Grad: Don, what is this time period that you're making those comments about?

Haderle: This was the late 1970s. In the late 1970s, we came to this realization, and then the early 1980s we were doing the engineering to go and fix it before we ended up releasing it.

Grad: Let me explore the IBM situation a little bit further. You're starting System R in the 1973, 1974 time period. You've got a high-quality group of people out here in San Jose and Santa Teresa. What was it called at that point?

Don Chamberlin: It was called the IBM San Jose Research Lab.

Grad: San Jose Research Lab. Okay. And you're starting this project. You've got a good-sized team. What are you attacking first? What is your first issue?

Implementation of Phase Zero and System R

Chamberlin: I'd like to answer that question by expanding on something that Mike Blasgen said about what we called Phase Zero. It turns out that in the history of the System R project, there were two completely separate systems that were built. The first one was called Phase Zero, which took roughly from 1973 to 1975. And then the second one, which became known as System R, took about from 1975 to 1979. And this came about because the group was originally organized into two parts. The first group, called the RDS, was oriented towards users and languages and SQL. And the lower-level group, called the RSS, was oriented towards access methods and disks and B-trees and transactions. And so we both began doing our thing. The RSS group began building index managers and storage managers and so on. The language group had a language but the RSS wasn't ready yet, and so we built our initial SQL phase zero prototype on something called XRAM. XRAM was a relational access method that had been developed by Raymond Lorie at IBM Cambridge. And so the first SQL implementation, which was described in a paper by Morton Astrahan in "Communications of the ACM," was an interpreter for SQL built on top of XRAM. And then when the RSS was ready and mature and able to be used as a platform, we built a second prototype on top of the RSS, and this was the point when Raymond Lorie had his key breakthrough, I think, that said rather than building an interpreter on top of the RSS like we did before, let's build a compiler on top of the RSS that compiles actual 370 machine code that calls the RSS access method to implement SQL queries. And I think that was the breakthrough that made it possible for the System R implementation to support online transaction processing where performance was very critical and where you did the same thing repeatedly. So you didn't want to interpret that thing. You wanted to compile it into some very efficient code that could be reused.

Moshe Zloof: So the concept was that on these queries, there is a reuse of the queries. These are not ad hoc queries?

Chamberlin: System R supported both ad hoc query and repetitive transactions.

Grad: But you were compiling in either case, though?

 © 2007 Computer History Museum

Chamberlin: We compiled in either case, and I think the compiling was especially important in the OLTP environment.

Zloof: Just a comment that QBE was also based on XRAM.

Grad: Oh, is that right?

Zloof: Yes, exactly. Raymond Lorie's system. And we, of course, just interpreted. We didn't compile.

Grad: I want to come back to that, because I want to make sure we cover QBE as part of this session, Moshe. That's important. Are there other comments now? System R is ongoing. Team size-- this is all still under Research?

Chamberlin: It was all under Research, and I believe the maximum number of people working on System R at one time was 20. And most of the time it was less than that.

Grad: Okay. I have a story that I've heard, and I promised I would bring it up here and ask those of you who know whether it is accurate. I was working in IBM Research in 1976 and 1977 under Joel Birnbaum. At the time, Joel was pushing very, very hard to get relational supported as an IBM product, and my job at Research was to help transition products out of the labs and into the product divisions. And in our case, it was DPD. I could never persuade the DSD at that time to do anything. DPD had done CICS. DSD had ignored us and wouldn't pick it up. We had done IMS. Again DSD ignored us and wouldn't pick it up. We had a half a dozen major products that were selling all the IBM hardware, and the Product Division said, "You guys don't know what you're doing." So that was at that period. So we thought we could take some of the things out of the Research Center, bring them directly over into DPD and avoid some of the headaches at the product divisions. Joel wanted to get relational out of the labs. He told me, or I think he told me at the time, that one of the IBM Fellows had come up with the idea that there could be a combined database management system that would handle all the different kinds of models-- whether you like the word hierarchical or network or what have you, but that this could handle relational and all the others and he was going to design and have it done. And this had stymied any release of any products, and that this continued for a couple of years. And I left IBM in 1978 and that it was not until 1979 that someone said, "It's not going to ever work. The hell with it. We have to get a product out here to compete with the other people who are now in the relational business." Now, that's the story I heard at the time. Does anybody know if there's any truth to that story?

Funding of a Relational Product

Sharon Codd: Well, I'm not sure about that. But I do know the event that caused IBM to put relational database into the product plan, that is, to fund it and to, you know, assign resources to it.

Grad: What was that?

Codd: It's kind of funny. There was a little company in Connecticut-- I think it was in the 1970s-- called National CSS.

Grad: Not such a little company. I know them well.

Codd: All right. They built a relational-like product. I don't know myself, you know, how relational it was, but it was a relational-like product that ran on IBM mainframes. And a number of large users who were looking for an ad hoc query product got wind of the National CSS product and bought it. I don't know the name of the product. I don't recall.

Grad: NOMAD is the only product they built during that time frame.

Codd: Well, then that must be it.

Grad: We have a study on them. They're in our database on the Corporate Histories Project. We have a major set of reports on them.

Codd: Well, it was this particular product that IBM customers were starting to buy. These were the large customers using IBM's mainframes. And typically to get this query work done, they'd put in another mainframe to do it so everything was wonderful. However, the National CSS product did not support IBM's multiprocessors, only the single processor. And so as the query product got more and more usage, the single processor ran out of juice and there was nothing to move to in the IBM hardware product line. Consequently, two large customers were unhappy--one was Bank of America that was using it, and the other was I believe an oil company also in San Francisco

Haderle: Chevron.

Codd: Okay. They went and bought Amdahl machines. Now when IBM lost a large mainframe, the loss report went all the way up to the top. And the top at that point was Frank Cary. And there was a whole discussion in the Corporate Technical Committee and with various other committees on what this new technology was all about, and the word got down to

 © 2007 Computer History Museum

Ted Codd. And as Ted tells the story, when Cary heard about this relational technology, he asked the question, "Is it ours?" In other words, are we the ones that came up with the idea? And when he heard the answer was yes, he made the verdict.

Grad: Do you have any idea of the time period?

Codd: I'm trying to think. It was some time in the 1970s.

Haderle: 1978 or 1979.

Codd: Yes. That's the time period.

Grad: That would not be inconsistent with what I heard. A man's name that was mentioned to me was Andy Heller as being the one who was doing this project.

Chamberlin: That's Eagle.

Codd: Yeah, that's Eagle.

Grad: What was the connection between Eagle and System R?

Lindsay: Well, I think Eagle might have been an attempt to cover all data models with one storage unit.

Codd: Yes.

Haderle: At the risk of taking way too long let me elaborate. We had Research inside of IBM, which had its own funding. And then we had product divisions, and at that time the product divisions-- Carl Conti ran the bulk of the stuff that produced revenue, which was the processors, along with the storage devices from General Products Division. By this time IMS had been out for some five or six years and they saw the relationship: when IMS was in there, customers bought lots of storage, and when it wasn't, they bought less storage. So, you know, databases are good; it produces demand for more storage.

Grad: That's what they're for.

Haderle: And so they funded the product divisions to go in and do something to do more on this database stuff--and you guys figure out what. And so the Eagle Project was formed. IMS had originally been a field developed program, responsive to a particular customer--

McDonnell Douglas [editor's note: it was actually at the Rockwell Space Division].

Grad: Then IMS became a regular program product.

Marilyn Bohl: Right. I was an IMS manager in GPD in '195.

Haderle: So this means it helps when a couple of customers want a product and then suddenly it gets wildly successful and you go, "Oh, my God." And you've got to fix up something that you hadn't prepared for having such an elaborate success. And so the Eagle Project was created to take the underpinnings of IMS (not the data model part), that included all the interactions with the operating systems having to do with logging, recovery and locking, storage management, etc. and go in and really fix all that up to make it robust. And that's what the Eagle funding was for.

Grad: GPD was funded to do that?

The Eagle Project

Haderle: No, The storage guys had the money, and they handed it over to the software guys, Gary Marx, and said, "You go and do more database stuff, whatever it happens to be-- that's your thing-- and generate more demand for storage." And Marx's team said, "Well, we have this IMS that's generating a lot of storage, and it's ad hoc but it can't scale so we need to do something to improve it." So they said, "Great, go and do that and show the improvement and make more storage and we'll give you more money." So that was the Eagle Project. It was to replace the underpinnings of IMS.

The Eagle people looked at what was going on in Research and, as Don Chamberlin pointed out, there was a part of System R, the RDS, the data model part, and there was the RSS, which was the underpinning of storage-- locking, logging, recovery, etc. The Eagle team borrowed a couple of RSS guys from the Research Division-- Franco Putzolu and Jim Gray-- to go in and try to improve these capabilities in IMS. Franco was interested in finding out if he could have a single underlying storage structure that he could have multiple models running on the same storage structure. And Chris Date came along with UDL which was a universal data language. UDL was designed to combine network as well as hierarchical as well as relational on top of this structure.

This was around 1977 or 1978. But by 1979, we hadn't produced anything that was running and the guys that gave us the money got a little nervous because they didn't see anything. And so there was a moment of decision. Mike Saranga was managing at the top and Bob Jolls was managing this project and they had to come up with a decision to do something, make a move

here. The decision was to dump the UDL thing and instead we'll take System R SQL and we'll put that out. And, in fact, we'll go to the VM market. IBM had two operating systems. They had more but two big ones. The primary one was MVS, which was for doing big transactional batch work, and VM, which was decision support, report writing. So the decision was to take this prototype here and commercialize it into the VM market for doing query decision reports. Then it was realized that it would make no sense after that to go and do some other kind of a different kind of a data language over on MVS for transaction support. So we decided to carry the RDS that we would have on VM and bring it over to MVS as well. This was around 1978, 1979, the decision.

Grad: RDS on top of the Eagle base.

Haderle: On top of the Eagle base; and the Eagle base was really on top of a restructured IMS. System R had been developed on top of the VM operating system, and the MVS operating system was different. MVS was targeted for 24x7. Let's say it tried to stay up for a shift.

<laughter>

Bohl: Well, but that was good because it was just evolving at the same time that you evolved and they put things in there.

Haderle: So it was more robust, intended for running your bank for eight, 10 hours a day, I mean, at that time.

Jerry Held: That was good. It let Tandem get started because we could run it for 24 hours a day.

Haderle: There you go. We left an opening for you.

Held: It was nice of you.

Haderle: Yes. Well, we needed to leave something.

SQL/DS and DB2

Ken Jacobs: You didn't actually ship SQL/DS until 1983 or so?

Haderle: Well, it was in limited availability in 1981, 1982. We kind of leaked it out. Then it became generally available in 1983.

Jacobs: So the other opening you gave was Oracle, which came out in 1979.

Haderle: Correct. The money that was coming in for development funding was coming in from the storage guys and the correlation was between the database and the fact that you made more storage and that's where the big funding came.

Jim Strickland: I'd like to add just a little piece, Don. When the decision was made to take it from Research and move it to the product divisions, there were two different efforts. At first, Santa Teresa thought we were going to do the DOS, the VM and the MVS versions of this thing. When it was brought over to Santa Teresa, it was pretty quickly decided that Endicott would pick up the VM version and they did. The picked up the RSS and the RDS and put that out on VM fairly quickly. They then did a DOS version, and that came out a little bit later. So Endicott, before the Santa Teresa Lab ever had a product out, I think, had both the VM and the DOS product out. The Endicott products were called SQL/DS.

Endicott produced products more quickly because they used both the RSS and the RDS. We at Santa Teresa ended up picking up just the RDS, and with great difficulty plugging it onto the Eagle base.

Haderle: SQL/DS focused on what Sharon was talking about, which was we didn't have something for query, like NOMAD or RAMIS, and so this was aimed at that particular market to be able to go in there more quickly against that market to release this thing more quickly.

Grad: Santa Teresa, then, had responsibility for the MVS version?

Strickland: Correct.

Grad: And that became DB2?

Strickland: For a short time it had responsibility for all versions but then two of them went to Endicott.

Grad: So I guess the question is, from a technical standpoint, is DB2 a direct child of System R or is it a different implementation? Is it an Eagle-based implementation?

Chamberlin: Half and half.

Lindsay: Just a few more details on platforms. In fact, System R was ported to MVS and was used on MVS by Boeing. Boeing was one of the two early adopters. We had three including Upjohn.

Chamberlin: Pratt and Whitney, and Boeing.

Lindsay: The other thing, a little more detail on the RDS, RSS porting is that the minute that System R moved to Endicott, the first thing they did is rewrite or literally translate the RDS, which was originally implemented in PL/1, into PL/S, which was IBM's system programming language. My main recollection is that the side effect of that translation is they changed all the good old eight-character function file names into really “informative” file names. In IBM, you always had to use the first three characters of the file name to represent the product, so you got five characters left to say something about the file. So they said, "This file is ARI0001. This next file is 002." I found that pretty annoying when I was working with that code. It was this translated to the PL/S version of RDS that then moved to be the starting point of the MVS version, called DB2.

Grad: Now let's to get to more general technical subjects, things that all of you had to try and resolve and handle. Let's try and deal with that now. What were the technical issues that you were having on Ingres? What was going on in the other places? What were you looking at? What were your problems?

Held: Well, just as a general comment about what was going on, I think that we heard that there was a lot going on at IBM, but it was all inside of IBM. This whole period, mid-1970s through 1983, was a very long labor, right? And the baby wasn't born until the end of this period, right?

Grad: Were you getting feedback? Were you getting reports? Were you getting information on IBM from other sources?

Held: Well, just to say what was going on in the rest of the business, the Ingres Project was continuing at Berkeley for awhile and then became a company.

Jacobs: Oracle got started in 1977 and released its first DBMS product in 1979. So in two years, Oracle went from nothing to a product.

Tandem and Teradata

Held: From an industry history perspective, one of the things that allowed the industry to get going-- remember back then IBM was so dominant in every aspect of computing it was very difficult to break into any part of the business-- so the long gestation period of DB2 or whatever it eventually became called allowed for these other projects to get going. Ingres was able to morph from a university project into a company and develop down its path. Oracle got started. And I think two others which you're not talking about too much here really had a pretty

big impact. One was Tandem, which started the database in 1976 and actually started shipping high performance relational product in the mid-1970s. And remember the big discussion all through the 1970s, during the entire decade, was can relational actually be used effectively to solve problems. There's this beautiful model. Everybody agreed that the model was wonderful. But during that whole decade, there was still a question of would it ever stand up to IMS for real applications, because IMS was the dominant-- IMS and VSAM, without the database on top-- ran all the commercial data processing. What we first proved at Tandem, and we were able to get some of the best people. Jim Gray, Franco Putzolu, Stu Schuster, for example, we assembled a phenomenal team and built a very high performance database solving problems of non-shared memory multiprocessors, five nines of reliability, distributed database and all of those things, and that was all for OLTP.

Haderle: Wait, what year?

Held: 1976 through 1979. The first product came out in 1976, and by 1979 and 1980 we were doing a distributed system.

Haderle: But Gray and Putzolu weren't over there in 1976.

Bohl: Right.

Held: They didn't come for a couple of years.

Zloof: You're talking about Encompass, right?

Held: It started with Encompass and Form and then NonStop SQL.

Zloof: Yeah, but NonStop SQL came in the 1980s.

Codd: Right.

Zloof: That's my point. SQL was 1980.

Held: But the beginnings of all of that was--

Codd: Encompass.

Held: The other technical thing that was going on was that this was all relational based and with a relational query language and with a transactional processing agent. The other thing

that I think was interesting is that a group out of Citibank, which was one of the first users of the Tandem product, spun out and they formed a company called Teradata. I don't know the exact date, but it was taking the same ideas and pushing the high performance query side of things. I think in terms of early technological innovation for real impact. Even today Teradata still has a pretty important place in the industry, since it was able to take a number of those ideas and create a real product that was shipping to real customers.

Grad: What were the technical things? We've ended up talking much more about the business evolution and so forth than about the technical things that made the difference? That's what I'm trying to at least cover somewhat in this meeting. For example, what the specific technical innovations did Ingres make in implementation or in how things were done or in how the optimization of queries, something of that sort, that gave it movement?

Codd: Well, I think Tandem and Teradata did something very, very important and that is they understood and brought to market products that capitalized on relational's ability to make use of parallel processing.

Grad: So the parallel processing was a key issue?

Codd: Yes.

Haderle: But that's in the1983, 1984 timeframe.

Grad: We're trying to talk about activities up to1983.

Held: Well, parts of it were.

Haderle: Well, he's talking about NonStop SQL and she's talking about Teradata. We're out in that timeframe. We're not in 1978, 1979.

Codd: No, it was in the 1970s. And Encompass was there too.

Held: In the 1970s we had the distributed query processing and the non-shared memory locking, all that stuff. We had non-shared memory multiprocessors, which is actually a big part of today's database world was in the 1970s. And all of the locking and query management associated with that. The SQL language was adopted when NonStop SQL came along. And that was a mistake that Ingres made, too, was in not seeing the power that IBM had in setting the standard. It was very clear and probably one of the most insightful things that Ellison did was to piggyback on the power of IBM and SQL.

Jacobs: Digital was another company that made that mistake. When they came out with RDB later, they didn't do SQL and there was a big sigh of relief at the competitors. When Digital made their RDB, it was a relational database, but they designed their own language for it. And it's actually now an Oracle product, as a matter of fact.

Oracle

Grad: Well, what was happening at Oracle from a technical standpoint during this period?

Jacobs: Jerry pointed out the high-end OLTP and data warehousing requirements and the technologies that supported it. I think what Oracle did well was that it made relational database popular and small. I mean, running on a PDP-11 and on a variety of operating systems and fitting on a PC later in 1984 and in 640K, it's kind of hard to believe now. We did some things that were innovative for particular customer requirements that still are not widely available, things like hierarchical query capability we called Connect By to be able to traverse the materials-type applications. But the first SQL that Oracle did was relatively complete with views and sub-queries and so forth, but it wasn't very reliable, it didn't run very fast, but it addressed that other market, that decision support market that IBM was targeting ultimately with SQL/ DS.

Grad: Yet my understanding is that the first customers that Oracle had were pretty large. They were the CIA and people like that.

Jacobs: That's right.

Grad: Did they give a damn about these small machines?

Jacobs: You know, I think so, yes. I remember visiting some of these military intelligence sites that I can't tell you about that were using them in the early years.

Grad: He just doesn't remember.

<laughter>

Jacobs: They made me erase my brain.

Codd: That's an oxymoron, military intelligence.

Jacobs: But yes, they were using them because they had pressing requirements they

couldn't realize without the Oracle software. And, you know, they needed to do the ad hoc queries, they needed to do things that they couldn't otherwise do.

Codd: Also, I believe that at the beginning the markets were departmental rather than the total large corporation.

Jacobs: Sure.

Codd: And in a departmental environment, they typically had the budget to buy a small machine.

Jacobs: Yes.

Codd: And so that's how that grew.

Grad: But these are not technical issues. And I've been trying to understand the technology. We've talked a lot about IBM, what were the things that the other companies were able to do more quickly or more effectively?

Optimization, Performance and Concurrency Control

Codd: Optimization.

Grad: None of you have mentioned optimization. That's something that was very important.

Jacobs: Oracle didn't really get around to query optimization until very late. We had a rule-based optimizer until our Version 7 product, so we didn't even have cost-based optimization until very, very late. But, you know, in the early 1980s we were fighting with Ingres versus QUEL, and a lot of the technical debate was which is a better language. And some of the issues that we had, it didn't really ultimately matter what the better language was but, you know, that was one of the big debates. And another big debate later in the 1980s, maybe '85, '86, was around distributed database.

Grad: That we'll come back to in the next session.

Jacobs: There were certainly performance issues. I think, as I recall in the later part of the 1980s maybe, it was SMPs were starting to come out and being able to scale well. SMPs were the first real performance issue. We had locking questions.

Grad: But now, these are questions of performance on minis? Primarily minis. We're not even debating the mainframe work.

Jacobs: Oracle's first versions were table-level lock-- well actually, the first versions were not table-level locking. They were not even atomic at the SQL level. They were atomic at the row level but were transactional, so it was kind of strange.

Lindsay: I believe in the late 1970s and early 1982s, there were, to my memory, two areas in which we debated a lot about the technology. One, of course, was SQL query optimization technology, whether cost-based or rule-based would make it.

And I would like to tell a little story. You should never work on SQL optimizers; it's the worst job in the world because you may make a change in the optimizer that improves 99 out of 100 queries. The 99 guys whose query you improved don't call you up and congratulate you. But the one guy who got hit by your change-- and it's impossible to change it without hurting somebody-- will call you up and say, "What have you done to me?" So this is a hopeless area to work in.

<laughter>

There was certainly a lot of debate and discussion and marketing brouhaha about whether a cost-based or model-based optimization query plan should be chosen. And as I mentioned at the very beginning, choosing the query plan was perhaps the biggest single technical innovation that made this stuff possible. Remember, before relational, all database or information management programmers told the DBMS how to do it. The decision of how to do it and the technology and mechanism to use was quite an issue for many, many years as cost-based versus model-based, rule-based optimization.

Grad: Let's put this on the record. What do we mean?

Lindsay: The cost-based mechanism tries to estimate how much CPU and how much I/O is it going to take to do the query this way, with a nest-loop join, using this index, with a merge join, doing a table scan and a sort here plus another sort and a table scan and then a merge, estimating the actual physical cost-- how many CPU instructions and how many I/O events are you going to cause. The rule-based optimization said, "No, we will use heuristics, which usually work pretty well, like access to smallest table first, then the next largest table, then the next largest table."

Chamberlin: That's a very good explanation.

Lindsay: And of course it gets really complicated when you add up all the alternatives.

The other area where I think there was a lot of active discussion and research was in the area of concurrency control, that is, concurrent with the System R work, I believe that serializability was put on a firm mathematical basis by Jim Gray's papers. Much of that technology actually came from IMS, but he made it popular. Also the recovery stuff came from IMS, and he made it accessible to the rest of us. There were certainly a lot of discussions about which products support serializability.

Serializability means that any application sees the database as though it was the only application running against the database at that time. If other applications were accessing the same or different rows of the same or different tables, you couldn't tell from your application this was happening except for possibly getting a deadlock and being told to go away because there's been a problem here.

Jacobs: I wanted to comment on this because in the very early days of Oracle, a decision was made to implement multi-version concurrency control where queries were able to run even though there were concurrent updates on the same data, and those queries were given a consistent view of data. And this actually was introduced in Oracle Version 3, which came out in 1983. And this was the version of Oracle that was re-written in C. The first versions had been written in PDP-11 assembly language. Version 3 was the portable version, but it really introduced the MVCC capability and it implemented it by reading from what was called a "before" image file that had the old version of the data. That actually served as a major differentiator for Oracle for well over a decade, and I think today it has become quite popular. It's interesting to me that IBM has still not adopted that. So I think you're right that concurrency was an important area. I think Jim Starkey also had something to do with it. He'll claim to have invented MVCC and did InterBase and now is at MySQL.

Ingres

Grad: What's going on at Ingres during this period of time from a technical standpoint?

Paul Butterworth: From a technical standpoint? Well, for me this would be the early 1980s since I got involved in 1980.

Grad: Okay. That's fine.

Butterworth: The major issues we were working on were very similar. Kind of at the top of the list was performance and I know we used to laugh because in the early days we didn't measure transactions per second but seconds per transaction.

Grad: <laughs> That's cute.

 © 2007 Computer History Museum

Butterworth: There were, of course, the concurrency issues, but the other big things that Ingres was working on were basically two items, both kind of ad hoc query interfaces to the systems using report writers, and application development tools to try to make the technology more accessible. We were working on all of those problems in the early 1980s.

Lindsay: Was that Larry Rowe on the report writer end?

Butterworth: That was Larry on the report writer and the query interfaces, and then, of course, Mike Stonebraker was helping us with the kind of internal issues, performance and optimization and things like that.

Lindsay: Is it true that the logo of Ingres was the turtle?

Butterworth: The turtle was associated with the research project in some way, shape or form.

Held: After my time.

Grad: Is this a technical issue, Bruce?

Lindsay: Well, it has to do with performance.

<laughter>

Query Rewrite

Chris Date: I'd like to go back to the optimization. In your abstract for this session you say, "This session will discuss the mathematical work and its significance and the reasons why this was thought to be a better structure." I always thought that one of the reasons that this was thought to be better was precisely because of optimizability. And in particular-- and because we have a formal model-- we know that we can take certain expressions in relational algebra or whatever and convert them into semantically equivalent, more efficient expressions. I don't think the products do this very much. I haven't heard it mentioned in this discussion so far, so really I'd like to ask the question. Do people see that as an interesting facet of the model? Is it important? Do you do it in the products and if not, why not?

Lindsay: Certainly I know with DB2 research much later than this, there was an enormous effort in what we call query rewrite, which is exactly what you are describing. DB2 does a semantically equivalent transformation so the query can be easier to optimize. But that's much later. It was almost into the late 1980s that that was really recognized as sort of distinct from optimization, although it turns out some rewrites that are losers but most of them are winners.

The other thing I'd like to mention for the late 1970s and early 1980s is there was an awful lot of theoretical work in normal forms; the biggest waste of time, it threw people off their game,<laughter> prevented them from looking where they should have been looking. When the fifth normal form came out, I gave up. It was not helping anybody and, in fact, our advice to customers was to do selective de-normalization because you need it for performance. So I think this was actually a red herring that actually put people on the wrong direction quite a lot.

Date: Actually I think that's a red herring.

<laughter>

Date: I want to stay with the optimization thing for a moment because query rewrite, yes, that's exactly what I'm talking about. There was a prototype at IBM which was operational in 1971. It was called IS1, Information Systems 1. And then for political reasons they changed the name to PRTV. This was designed to be a name that was deliberately unmemorable. That stood for Peterlee Relational Test Vehicle. That was operational in 1972. But what they did in optimization was query rewrite. That's what they did and they did all kinds of interesting things.

Grad: This was in the U.K. in the early 1970s?

Date: In the U.K. And I've never understood why that work was not picked up by IBM's product people.

Grad: Did any of the U.K. products ever get picked up in IBM as IBM products?

Date: CICS.

Grad: Until Hursley took over CICS from the U.S., they never had one of the major products to my knowledge.

Date: They had PL/1 before that, when I was there.

Grad: Oh, PL/1. That's a major product?

Date: It was then. It was making as much money as COBOL in those days.

Grad: But COBOL was being given away for free.

<laughter>

 © 2007 Computer History Museum

Butterworth: Just a note on the query rewrite stuff from the Ingres perspective, Ingres always had a little bit of query rewriting in it. It must have been 1982, 1983, when we put a new query optimizer, into the database system. It really had two big functions. One was it did do a lot of query rewriting, and the second one is that its cost estimates were based on a set of very detailed statistics that it had captured about the distribution of values in the tables. And when we put that in, that made a tremendous difference.

Lindsay: In which direction?

Butterworth: Well, it helped most queries. A few, though, got away.

Grad: There was a concept of adaptive learning in dealing with queries, repeat kinds of things, and the idea of keeping track of what had happened before and then to guide you in how you would handle queries in the future. Was any of that done at the time?

Jacobs: Later.

Lindsay: Automatically in the product? We've been talking about it for 50 years.

Jacobs: I believe DB2 had this Leo Project. I don't know the details. Oracle's current release has some of that. But that's way out of our timeframe.

Grad: I wrote a thing called decision tables 50 years ago and more, and one of the things the IBM Research people under Ralph Gomory thought was so great is that we could track what was happening and then we could determine which columns we wanted to do in the order that would give us the most likelihood of success and measure how much each of the queries was going to cost us. And this was back in the early 1960s.

Other Technology Developments

Michael Mahoney: Can I ask a question? As this research is going on, what other areas, if any, are you drawing ideas from? Are ideas coming out of the programming languages community or other research communities, or was this all fairly homegrown, especially these more theoretical and conceptual issues?

Grad: I guess, Chris, was there more research going on at this point that was being published?

Date: I was never in research myself so I'm not really the right person to answer it.

Grad: Were there things being published that you were aware of? You were writing the books. Everybody was reading your books. How many hundreds of thousands? I heard on that one book it was 700,000, is that the right number?

Date: Somewhere over that.

Grad: That's a hell of a number.

Held: But it is true that in the first few years in that period, 1973 to 1975 or 1976, when all this activity started, there was an awful lot of thought and publications on ideas, anything from referential integrity, security constraints or optimization. A whole list of ideas that were sort of laid out as possibilities in papers, and then it was sort of the next decade where people went and tried to implement a lot of those. Some of those things didn't see the light of day for a decade, but if you go back and look at all the papers we wrote between IBM and Berkeley and other places, there was a broad, broad set of ideas that were sort of laid out and then it was at least 10 or 15 years to go out and implement them all.

Grad: Were there any significant developments on either SQL or QUEL during that time period from the early 1970s to 1983 that made a major technical difference?

Strickland: There certainly was one. In the IMS hierarchical system the parent-child relationship was very important. The parent-child relationship isn't a concept in the relational model and therefore if you do a join of parents with children and there are no children, you get no parents. The missing parent was really important to the IMS community and so when we discovered, late in the game, that a join did not give you parents without children, it was a big blow to the effort to build a single database that would have both SQL interface and DL/1 interface. That was a really technical step in my mind, a problem that caused IBM to split and say, "Okay, IMS forever. DB2 will be in a separate box."

Grad: That's an important point.

Jacobs: Well, but Oracle had an outer join in the very early days. I think even in Version 2, our first commercial implementation, had an outer join.

Lindsay: Everything Ken mentioned.

Grad: So the outer join was the solution.

Lindsay: Preserving the childless parents.

Jacobs: But I have a question I've always wondered about and it is why did the evolution of database separate from operating systems. Why didn't we make operating systems with transactional data stores and searching and so forth? It's always struck me as odd that we didn't endow our operating systems with the same rigor and robustness that databases provide. And I think we've tried over the years to see these things come together, but why was that at IBM?

Lindsay: Both Jim Gray and, I think to a large extent, Stonebraker, took the point of view of get out of my way, operating system.

Jacobs: But in a sense, the database should be the operating system.

Lindsay: And you might recall a paper by Jim Gray called "Database Operating Systems."

Strickland: And IBM's answer, I think, would be partly geographical and partly hardware. The geographical part was 3,000 miles. The operating system was in New York and the database was in San Jose. And the operating system was built to optimize the hardware. That was their role. Other stuff was unimportant.

Grad: You have to remember where some of these came. There's a little history. Remember, it ends up that IMS and CICS, data base/data communication, was totally a product of the sales division of IBM (the Data Processing Division) built in response to specific customers who needed specific things. Utilities companies, in the one case, the aerospace companies in the other. We could not get the product divisions to pick it up, so we have many years of development, with lots of hardware being sold as a result-- and remember, the 1970s were a terrible time at IBM in terms of selling new hardware. The only thing that was driving it primarily was these DB/DC systems which were making a hell of a difference. Relational comes in, at least at IBM, in this whole pattern. By this point you have six major database management companies, very successful software companies out there competing. You mentioned earlier the Software AG and Datacom/DB at ADR. They're big companies. Cullinane comes in with the IDMS product. These are big companies taking major market from IBM. CICS has totally picked up the DC market. You make the same argument about online transaction processing. So I think as relational develops, it develops as a competitive framework to some extent, doesn't it, to what's already established as a separate business.

Haderle: Small point of correction: IMS was successful by this point in the 1970s, and again, IBM is managed by hardware, right, selling processors and storage. The changes to the operating system are really to sell more processors and storage. If you wanted to get changes in dispatcher and you were IMS, you could get changes. But if you were relational, where you had no customers, you didn't get changes. So you had to build a little something on top of it to be able to get something more lightweight, right? Your own little dispatcher. And CICS was the

same way. CICS came out in 1976, there, and they didn't have any customers. By the time they were commercially successful enough to get the operating system to acknowledge them, it was around 1975, 1976. Prior to that, they didn't have any acknowledgement. For us, on the relational side, to get the operating system to go in and put in some lightweight primitives down there was impossible because we had no customers. We were forced to build a system on top of a system. Later on you can go and try to push that stuff down, but at the time, you're the new kid on the block. It was business, not really a technical decision. [Editor's note: CICS was released as a product in 1969 and had as large a customer base as IMS in the early to mid 1970s].

Lindsay: Don, was there any consideration at this point of the fact that we gave the operating systems away, but we were beginning to charge money for the applications?

Haderle: No. Again, in the 1970s, the software was not a separate business. It was there to sell the storage, and the fact that it drives some revenue was, well, exciting for somebody but it really was not going to pay for itself in the 1970s. [Editor's note: Software was a separate business in IBM starting in 1970].

Grad: We had set the line to separate operating systems from all the rest of the programs, but by the end of the 1970s, that line was starting to dissipate. We were going to sell operating systems at IBM. But the point is it's a much bigger context by now.

Haderle: Yes, we were selling operating systems. We were getting, you know, a penny off of the operating system, and off of the hardware we were getting, you know, thousands of dollars. So it was like, you know, you were getting chump change at the end so this was not driving where the business was going.

Blasgen: This is not technical. I will make a technical comment if you like, but Jim Frame actually pointed out that IMS DB alone was licensing at $1,500 a month. It's not a trivial. Maybe it wasn't enough to pay for all the programmers in Palo Alto that we eventually moved to Santa Teresa, but it was still a significant revenue source, and it was a big problem for relational because forgetting about whether or not relational would sell, it certainly looked like it would upset $1,500 revenue per month per installation.

Grad: That's a problem.

Blasgen: And IBM definitely did not like that.

Haderle: If you want to go back to our technical issues, after the research we had the model. In 1976, 1977, 1978, our problems then in terms of commercializing it into the

mainframe/batch market were manageability, concurrency, running against SMPs, performance, and being able to compete against the other guys in terms of number of MIPS in cycles and storage sectors and whatever. The biggest issue going into transaction processing, the sweet spot in the business, was manageability, keeping it up for multiple shifts, and to be able to handle concurrency. Concurrency control was a huge issue for us, a huge issue. A row-level locking button didn't exist at that time.

A killer for us was not having referential integrity in the product. They would not move from any transaction database to this one without all the integrity constraints there at the outset. So while we were taking the query compiler, etc., and doing that, I'd say there was this research thing going on in terms of how you improved that. But commercializing it-- which took multiple years-- required making it commercially robust for performance, availability, manageability, etc. There are lots of technology issues in there.

Strickland: A major technical issue is the user-friendliness of the system. We haven't discussed that. IBM did not do as well as I might have hoped. We kept trying at that, and did fairly well, I guess, with VS/QUERY. I'm not too familiar with it. But the point is, that's a technical problem that we haven't covered.

Grad: We have another session tomorrow morning on technical, and we'll pick up there. Actually, we didn't get a chance to pick up on the Sybase stuff so we'll pick that up in the early 1980s and then go from there forward at that point in time. A lot of stuff happens in the 1980s technically, a lot of changes, a lot of improvement.

Don Haderle's point is absolutely correct. I could get no traction on my software products at IBM because it was hardware, hardware, hardware decisions. Only to the extent I could support hardware would they let me sell anything. I had CICS. They were fighting me tooth and nail, and I was bringing in millions and tens of millions of dollars. It just galled me that it was a hardware-driven decision.

© 2007 Computer History Museum