FUTURE DIRECTIONS IN DBMS RESEARCH

*The Laguna Beach Participants**

Abstract

On February 4-5, 1988, the International Computer Science Institute sponsored a two day workshop at which 16 senior members of the data base research community discussed future research topics in the DBMS area. This paper summarizes the discussion which took place.

1. INTRODUCTION

A computer science research laboratory has been set up at the University of California, Berkeley called the International Computer Science Institute (ICSI). ICSI has been sponsored and has received its initial support through the German National Research Center for Computer Science (Gesellschaft fuer Mathematik und Datenverarbeitung - GMD). In addition to pursuing a research program in fundamental computer science, ICSI has the mandate to promote the interchange of ideas and the collaboration between computer scientists in the US and in other parts of the world. As such, ICSI sponsored a two day workshop in Laguna Beach, California, attended by 7 senior DBMS researchers from Germany and 9 from the USA. This workshop was organized by Erich Neuhold of GMD and Michael Stonebraker of Berkeley.

The purpose of the workshop was to discuss what DBMS topics deserve research attention in the future. During the first day, each participant presented four topics that he was **not** working on that he thought were important and that he would like to investigate. In addition, each participant was asked to present two topics that others were working on, which he thought were unlikely to yield significant research progress. All participants then cast five votes in support of research topics proposed by others. They were also given two votes to indicate agreement with overrated topics. In addition to the voting, time on the first day was devoted to ''war stories'', i.e. experiences of the participants in working on very hard real-world data base problems. The second day of the workshop was devoted to 40 minute discussions on a collection of specified controversial topics.

The discussion was far reaching and significant agreement was present on a number of issues. For example, the participants were nearly unanimous on the importance of research on user interfaces, active data bases and parallelism. All other topics received noticeably less support. The participants were unanimously negative on the prospective research contribution of hardware data base machines, general recursive query processing, and interfaces between a DBMS and Prolog. On other issues such as the importance of research into data base tool kits, the participants held divergent positions.

This report summarizes the discussion which took place. All participants contributed to its preparation and have approved its content.

The remainder of the report is organized as follows. Section 2 discusses the applications that the participants thought would drive future requirements of DBMSs. Then, Section 3 considers the hardware environment in which DBMSs will be required to run in the future and discusses the viability of hardware data base machines. In Section 4, the relationship of a DBMS with other system software including the operating system and language processors is treated. Section

* The Laguna Beach Participants were Philip A. Bernstein, Umeshwar Dayal, David J. DeWitt, Dieter Gawlick, Jim Gray, Matthias Jarke, Bruce G. Lindsay, Peter C. Lockemann, David Maier, Erich J. Neuhold, Andreas Reuter, Lawrence A. Rowe, Hans J. Schek, Joachim W. Schmidt, Michael Schrefl, and Michael Stonebraker.

5 turns to mechanisms for DBMS extensions including so-called object-oriented data bases. Active data base management systems are treated in Section 6 along with the attitude of the participants toward rule systems in general. Research in end user interfaces and application development tools is the subject of Section 7. Future research in techniques for implementing single-site DBMSs is considered in Section 8, while Section 9 turns to distributed data base management systems. Lastly, Section 10 consists of a collection of topics which did not fit well into one of the previous sections.

2. FUTURE APPLICATIONS

Several participants suggested investigating the needs of a specific application area as a fruitful research endeavor. Others used new application areas as examples to motivate the discussion of specific research topics or in exploring specific issues. Lastly, many of the war stories concerned new application areas. This section summarizes the themes that emerged.

2.1. CASE

Several participants pointed to Computer Aided Software Engineering (CASE) as a significant new application area for DBMS technology. Basically, all information associated with a computer program (source code, documentation, program design, etc.) should go into a data base. The participants thought that the needs of clients in this area were similar to many other engineering applications (e.g. versions, complex objects, inheritance, etc). Some current commercial systems are using data base management systems to manage CASE data while others are building custom DBMSs with needed capabilities. There was substantial agreement that CASE is an important application areas with meaty problems worthy of investigation. For example, a good way to build a generalized ''make'' facility (i.e. one which would automatically recompile appropriate software modules when changes occur) would be to support a powerful trigger facility. As noted above, research on active data bases received nearly unanimous support from the participants.

2.2. CIM

A smaller number of participants pointed to Computer Integrated Manufacturing (CIM) as a significant new application area. Here, data on all phases of plant operation would be stored in a data base system (e.g. code for machine tools, test results, production schedules, etc.) Participants pointed to new data base capabilities that were needed to support applications in this area such as alerters and triggers. Considerable support for working on the problems in this area also emerged.

2.3. Images

Two of the participants pointed to image applications such as found in medical diagnosis, natural resource management (e.g. satellite data) as an important research area. The discussion pointed out the need for storing large (several megabyte) bit strings of images in a DBMS. Beyond storing large objects, the participants felt there was a substantial pattern recognition problem in this area that should be addressed by others. Nobody thought it was hard to put very large but relatively unstructured objects efficiently into data bases.

2.4. Spatial Data Bases

Several participants used spatial examples as hard applications. These were typically geographical data bases containing encoded information currently found on maps. The problems varied from providing urban services (e.g. how do I get from X to Y efficiently on public transportation) to conducting a military operation to environmental information systems integrating all kinds of data from under, on, and over the earth surface. There was widespread support for the importance of such applications, and the participants generally thought that this was a very good application area for extendible DBMSs.

2.5. IR

One participant thought it was important to investigate the needs of information retrieval applications. Basically, he thought that text would be a component in many data base applications and that the key-word oriented searching prevalent in today's text and information retrieval applications should be reconsidered. The objective would be to efficiently support the needs of clients in this area with semantic and object oriented models so that IR systems would not have to be written as "one-off" applications and the power of generalized DBMSs would come to bear in this field. Some interest for this point of view emerged.

3. THE FUTURE HARDWARE ENVIRONMENT

3.1. The Revolution

Several participants pointed out the there appears to be a "knee of technology" that is occurring. Specifically, individual high end CPUs have increased in speed from about 1 MIP to about 15 MIPS in the last two decades at a consistent rate of about a factor of two every five years. For the next several years RISC technology will allow CPUs to get faster at about a factor of two every 1-2 years. This factor of 2 1/2 - 5 acceleration in the growth rate of CPU technology is likely to have a dramatic impact on the the way we think about DBMSs, operating systems and user interfaces. The participants discussed the machine of 1991 as having 50 MIPS and a gigabyte of main memory, selling for $100,000 or less. Semi-serious comments were bandied about such as "if your data base doesn't fit in main memory, wait a few months and it will."

There was a wide-ranging discussion on whether the application needs of clients would increase at least as fast as the price of resources (CPUs, memory, disks) will be driven down over the next several years. The cost of an application with a constant need for resources will decrease by something approaching a factor of two per year for the next several years as noted above. Clearly, a client with a constant application will solve it with ever cheaper hardware, and performance will become uninteresting sooner or later. In order for data base management systems to continue as a vital research area it is necessary for the needs of clients to increase at least as fast as the hardware is getting cheaper.

There was substantial agreement that DBMS clients have large unmet needs that can take full advantage of cheaper resources. Several participants reported that decision support applications were frequently not being done because they were currently too expensive. Clearly, cheaper hardware will enable this class of applications. Others pointed out that clients justify substantially increased resource consumption on the grounds of human productivity. For example, they would use additional hardware to create real time responses to user interactions rather than in batch. Several participants pointed out that real world data bases are currently growing at 30-50 percent per year and the growth rate shows no signs of decreasing. Hence, space consumption is increasing at about the rate that disks are declining in price. Another participant noted the DBMS functionality would increase by at least the rate of decrease of hardware costs, and therefore performance of the DBMS will continue to be an issue. Lastly, one participant pointed to the scientific community who will consume any amount of available resources to refine the granularity at which they keep information.

The conclusion was that carefully managing resources applied to accessing data would be a significant problem for the foreseeable future. The only impediment to this scenario would be the inability of the DBMS community or their clients to write new software for enhanced function and new applications fast enough.

3.2. Data Base Machines

There was overwhelming sentiment that research on hardware data base machines was unlikely to produce significant results. The basic point several participants made was that general purpose CPUs are falling in price so fast that one should do DBMS functions in software on a general purpose machine rather than on custom designed hardware. Put differently, nobody was optimistic that custom hardware could compete with general purpose CPUs any time in the near future.

4. THE FUTURE SOFTWARE ENVIRONMENT

4.1. Operating Systems

Operating systems are much maligned by data base researchers for providing the wrong kinds of services, especially in the area of file management. The informed consensus of the participants was that this problem is not likely to go away any time soon. It was felt that operating system designers would have their hands full for the next several years with networking software and substantial kernel modifications caused by the large increase in CPU speed and amount of memory that will be present in future machines. These new communication protocols will include ISO and special purpose protocols such as NFS. In addition, fast restart, non-stop operation, failure management and restructuring the operating system to have a small kernel and a layered collection of services on top of it are likely to consume the attention of the operating system designers for the forseeable future.

Any hope that OSs will get more responsive to the needs of the DBMS are probably optimistic. The only area where there is hope is in transaction management. Several participants pointed out the need for transactions which spanned subsystems (e.g. two different data managers or the data manager and the network manager). Such needs can only be addressed by transaction support built into the kernel or at least kept outside the DBMS. A minority position was that this class of problems is naturally addressed as a heterogeneous distributed data base system problem. This solution clearly is only acceptable if all need for transaction behavior can be kept inside the DBMS environment.

Several people pointed out that OS support for data bases was an important research topic. One participant discussed an application which required 10 mbyte per second access to his data. Such performance is only attainable by striping data over a number of disks and reading them in parallel. There was support for additional research on high performance file systems for DBMS applications.

The discussion also focused on whether current operating system interfaces (e.g. MVS) would be preserved. Everybody thought that we were stuck with current interfaces for a long time, just as our clients are stuck with SQL.

4.2. Programming Language Interfaces

There was a discussion of the difficulty of interfacing current DBMSs to current programming languages. Some participants felt that it was important to clean up the current pretty awful SQL interface for both embedded and dynamic queries. Others said that most users would be coding in application development languages (4GLs) and that this level would be hidden, and thereby it would be less important. The consensus was that there would be a lot of programs written in normal programming languages that made calls on data base services, and that a better interface would be a worthwhile area of investigation. Moreover, somebody has to write the 4GL interpreters and compilers and this person should see a cleaner interface.

The possibility of a standard 4GL was briefly discussed. Nobody expressed any interest on working on this essentially political problem. Any 4GL standard is expected to be many years away.

4.3. Prolog

There is considerable current research activity in interfacing Prolog to a DBMS. Many participants pointed to this area as one unlikely to have a significant impact. They felt that too many research groups were doing essentially the same thing. Moreover, most felt that the real problem was to integrate some portions of logic and rules systems into a data manager. As such figuring out efficient ''glue'' to interface an external rule manager to a DBMS is not very interesting.

5. EXTENDIBLE DATA MANAGERS

Topics in this area were revisited many times during the two day workshop. The participants pointed out that CASE, CIM and spatial applications need new kinds of objects. Moreover, there is no agreement in these areas on a small set of primitive object types, and extendible

DBMSs seem the only solution to these application needs. The discussion naturally divided into extension architectures and object-oriented data bases, and we treat each area in turn.

5.1. Extension Architectures

There was substantial controversy in this area. Some favored building a complete DBMS containing a parser, query optimizer, access methods, transaction management, etc. and then supporting user extensions within the context of a full-function DBMS. This approach is being taken, for example, in POSTGRES, PROBE, STARBURST, and VODAK. Others favored a tool kit approach where an erector set of modules would be provided to allow a sophisticated application designer to put together a custom system. This approach is being investigated, for example, by DASDB, GENESIS, and EXODUS. The discussion was heated between these options and we summarize some of the points below.

The advocates of the tool kit approach stressed that a sophisticated user could install his own concurrency control or recovery modules from perhaps several that were available in the tool kit. In this way, a tailored transaction manager could be built. They also pointed to application areas like CASE where a "lean and mean" system could be built containing only required capabilities.

The advocates of extendible systems argued that transaction management code is among the hardest to make fast and bug free. Hence, it should be done once by the DBMS super-wizard, and it is unrealistic for even sophisticated users to expect to do better than the super-wizard can do. Moreover, performance knobs on the transaction manager should be installed by the super-wizard if they are needed. The second point argued by the advocates of extendible systems was that most applications seemed to require a full-function system with a query language, optimizer and general purpose run-time system. If a user needs a full-function system, then the tool kit approach is not appropriate.

The participants were about equally divided between the merits of these two points of view. Moreover, one participant pointed out that there is a spectrum of extensions, and that the tool kit and extendible systems are merely at different places in a continuum of possibilities. For example, extendible systems disallow replacing the query language with a different one while allowing the inclusion or replacement of data types. The tool kit approach, on the other hand, offers both kinds of customization. Consequently, the tool kit offers a superset of the extension possibilities of extendible systems.

Clearly, the architecture of extendible systems is a good research area. Moreover, several participants discussed the importance of research in optimizing extendible data base systems. Hence, research on query optimizers and run time systems which do not have hard coded knowledge of the low-level access paths or join algorithms was widely believed to be important. Several participants pointed at the advantage of a distinction between the design environment, e.g. for configuring the extendible DBMS, for evaluating schemas and for turning performance knobs, and the usage environment where the complexities of the extendible systems are compiled and optimized away. Both environments, however, should be reflected in a single architecture and system.

5.2. Object-Oriented Data Bases

There was a long discussion on this topic. Some of the participants thought this whole area was misguided, while others thought it was a highly significant research area. In fact, virtually all participants held a strong opinion one way or the other on this research area. About half thought it held no promise of significant results while the other half thought it was a area where major results could be expected. This seeming contradiction can be explained by the fact that the proponents and detractors invariably discovered they were talking about different capabilities when they used the words "object-oriented data base" (OODB).

Hence, it is clear that the term OODB does not have a common definition to the research community. The participants lamented the fact that there does not seem to be a visible spokesperson who can coerce the researchers in this area into using a common set of terms and defining a common goal that they are hoping to achieve. This situation was contrasted with the early 1970s when Ted Codd defined the relational model nearly singlehandedly.

The proponents of OODB research made two major points. First, they argued that an OODB was an appropriate framework in which to construct an extendible DBMS. Researchers in this area are constructing systems which allow a user to write his own types and operations using the data language of the OODB itself. One goal of the OODB community is thereby similar to the goal of the extendible data management advocates. However, in these databases extensions are frequently accomplished through writing new data types and operations in the implementation language of the OODB and not in its data language. Some of the OODB proponents, however, pointed out that a OODB should be capable of invoking and linking to procedures written in a variety of programming languages making it again more similar to extendible data managers. In addition it can be argued that the object-oriented paradigm provides a framework where constraints on the defined operations can be enforced and managed. This will avoid all the problems that arise when extendible systems allow uncontrolled operational specifications.

The second major point was that inheritance is considered central to an OODB. A discussion of inheritance centered around the question of why it has to be so complex. It was felt that a good idea was needed to coalesce inheritance into something understandable to an average application programmer. Today different kinds of inheritance are frequently defined only via examples and have to be expressed using the same specification mechanics of the data language. In multiperson systems this leads to all kinds of semantic confusion and handling errors.

The opponents of OODB research made two points. First, they said that many decision support applications are largely built around the paradigm of browsing data collections constructed in ad-hoc ways. Hence, it is important to be able to easily do unstructured joins between disparate objects on any conditions that might be of interest. Systems which do not allow value-based joins between all types of objects are thereby uninteresting for decision support. Consequently, most (but not all) systems currently advertised as OODBs are uninteresting because they do not allow ad-hoc joins.

The second point made by the detractors is that many researchers use the term OODB to mean a storage manager which supports the notion of objects that can have fields which contain identifiers of other objects and which performs the following tasks:

given an object-identifier, fetch the object instance
given an object instance, find any field in the object

This level of interface requires a programmer to obtain an object identifier and then "navigate" to other objects by obtaining object identifiers that are fields in the object. This reminded several participants of CODASYL systems, whose severe problems were discussed at great length in the mid 1970s. This class of interfaces was consequently seen by some as a throw back to the 1970s and a misguided effort.

These positions are not really in conflict with each other. Most of the proponents of object-oriented data bases did not defend record-at-a-time navigation. Only a couple of participants felt there were application areas where this interface might be appropriate. All thought that an OODB should have a query language with value-based joins or other means of set-at-a time linking operations along with a substantial data manager to process such queries. The OODB opponents generally thought inheritance was a good idea, and everybody was in favor of extensions.

Hençe, the real problem is that "object-oriented" means too many different things. Until the terminology stabilizes and some coherency of system goals emerges, we fear that others will consume much discussion time, as we did, in realizing that we didn't have a common meaning of the terms.

6. ACTIVE DATA BASES AND RULE SYSTEMS

Many participants pointed out the need for so-called **active** data bases. Hence, triggers, alerters, constraints, etc. should be supported by the DBMS as services. This capability was one of the needs most frequently mentioned as a requirement of future applications of data bases. There was extremely strong consensus that this was a fertile research area and that providing this service would be a major win in future systems.

Several participants noted that current commercial systems are installing procedures as data base objects which can be executed under DBMS control. In many cases such procedures are directly invoked by an application program. It is only marginally more complex to allow such procedures to be invoked as a result of conditions in the data base. Hence, active procedures and active data bases should probably be considered together as a single service. The participants noted a clear need for researchers to take the lead in defining better syntax for procedures, because current commercial systems seem to do a poor job of syntax and semantics in this area.

There was also unanimity that an active data base system should have a fairly simple rules system that would have extremely high performance. Questions normally addressed by AI researchers under the rubric of expert systems, (e.g. implementing a theorem prover to prove safety of a rule or mutual consistency of a rule set) should be completely avoided. One participant pointed out that a DBMS could simply fire rules at will and remember DBMS states as rules were activated. If the run time system ever returned to the same state again, then inconsistency or unsafety was present. In such a case the DBMS need only abort the current transaction to back out of the situation. Simple solutions such as this were universally preferred by the participants to attempts at theorem provers.

There is substantial current research activity in efficiently solving general recursive queries that might be posed by a user or result from activating a recursive rule. The participants felt that there are a number of clients with parts explosion or other transitive closure queries. There are also a number of clients with linear recursive queries such as shortest path problems in graphs. However, none of the participants had seen an application that was best solved by coding a general rule set and then optimizing it. As such, it is not clear that there is a client for solutions to general recursive queries.

An analogy was drawn to dependency theory which was explored at length a few years ago primarily by the theoretical research community. Most participants felt that little of practical significance had been contributed by this lengthy research beyond the original introduction of the theory, and that general recursive query processing would meet the same fate.

One participant noted that knowledge acquisition was a central and difficult problem in rule based applications. The sentiment of the workshop was that this task is extremely difficult. A small number of participants felt that significant advances would result from research in this area.

7. END USER INTERFACES

The participants noted that there are virtually **no** researchers investigating better end user interfaces to data bases or better database application development tools. In the last few years there have been only a scattering of papers published in this area. Moreover, there was universal consensus that this was an extremely important area, and it received more support from the participants than any other area. In addition, several participants noted that major advances have been made in non-database related user interfaces in the last decade. They noted that spreadsheets, WYSIWYG interfaces, Hypercard, etc. have caused a complete revolution in human interfaces.

The discussion turned to the obvious question "Why has major progress been made in a vitally important area with no participation from the database research community and what can be done to change the situation?" Several points emerged from the discussion. First, it was noted that publishing papers on user interfaces is inherently difficult. As one participant noted ''The best user interface requires no manual.'' Hence, how can one write papers that capture the essence of such interfaces? A second point was that the academic community (especially program committees) is hostile to user interface papers because they typically have no equations or ''hard intellectual'' content. A third point was that user interface contributions must be built to assess their merits. One participant advocated ''demo or die'' to prove an idea. However, to build real systems there is a mammoth amount of low level support code which must be constructed before the actual interface can be built. This ''infrastructure'' has only recently become available in the form of toolkits such as X11.

The participants noted that because of the last point it is now easier to work on user interfaces and there may be more work in this area in the future. Almost one-third of them said they were working on end-user interfaces of one sort or another. A constructive comment which emerged from the discussion was that international meetings such as SIGMOD and VLDB should set up a **video** track, through which researchers could contribute an idea in video rather than paper form. Such a stream with its own program committee would clearly be oriented to user interface contributions and has been used successfully by recent OOPSLA conferences. Collectively, we would strongly encourage the organizers of such meetings to implement this suggestion.

8. SINGLE SITE DBMS TECHNOLOGY

The undertone of discussions on this topic was that technology is in the process of changing the rules. Contributing factors are the imminent arrival of increasingly high performance cheap processors sometimes in shared memory configurations, gigabytes of main memory, and large arrays of 3 1/2'' or 5 1/4'' drives. This technology will require rethinking the query optimizer, execution techniques and run time system for the new environment.

A second point made by several participants was the need for high availability and much better failure management. Nobody thought that problems in this area would be easy because significant results might entail considerable low level ''nuts and bolts'' engineering.

There was widespread consensus that concurrency control for the traditional database transaction model was well understood and that papers in this area tended to be ''epsilon variants'' on previous work. Consequently, the participants were pessimistic that significant results would be forthcoming in the future. On the other hand, there was great enthusiasm for work on new transaction models. The standard notion of transactions as serializable and atomic has not changed in 15 years. There have been occasional wrinkles such as nested transactions and sagas. However, the sense of the meeting was that new application areas such as CASE and CIM would benefit from a more comprehensive transaction model. Such a model might encompass

1) check-in check-out protocols with version control,
2) weaker semantics than serializability or
3) transactions which can be UNDONE after committing

There was no consensus on what the model would look like, just that it was needed. The participants were vocal in their support for research in this area.

Strong support was also voiced for research on parallelism. Basically, this amounts to running multiple query plans for a single query language command. This tactic is valuable in a single CPU system in order to keep multiple disks busy. In a shared memory system it is desirable to utilize the multiple processors that are present, and obviously it is a central idea in tightly coupled distributed systems. The big benefit from research in this area is the ability to give a user the illusion of real time response as well as the ability to hold locks for shorter periods of time. Next to user interface research and active data bases, this topic received the most support.

There was some interest in research on tricks to achieve better performance. Work on removing hot spots, caching the answers to queries, and precomputed joins of one sort or another were mentioned as interesting areas. Each achieved some interest but was considered a limited scope idea, (i.e. something that a few people should probably look into).

Lastly, there were participants who were very hostile to research on variations of extendible hashing and B-tree access methods. The sentiment was that the recent papers are exploiting the same collection of themes, and that only a 10-20 percent increase in performance is likely to result from such ideas. A couple of participants noted that several commercial vendors have rewritten their data base systems in recent years and have not installed extendible hashing or enhanced B-tree schemes. The apparent reason is that they do not see the cost-benefit of writing code in this area. However, some participants noted that research on new access methods to support new classes of objects (e.g. spatial ones) is needed.

9. DISTRIBUTED DBMS

The participants felt that distributed data base management systems had been investigated at length by the research community over the last ten years. Now there is intense commercial activity in this area, and several vendors are hard at work on heterogeneous (or federated) distributed DBMSs. Moreover, the really hard problems in this area center around system administration of a large (say 50,000 terminal) distributed computing system. Managing this environment (e.g. installing new updates of modules, new users) is a major nightmare on current systems. It was felt that researchers probably have little to contribute to the solution to this problem because one must have experience with the needs in this area in order to make a contribution. Few researchers are in a position to closely study large complex systems of this sort.

The one area of distributed data bases that evoked support was the problem of scale. Clearly, data bases will be set up which involve hundreds or even thousands of nodes. CIM environments will have large numbers of nodes as well as environments where workstation data bases are present. There was support for rethinking algorithms for query processing, copies, transaction management, and crash recovery from the point of view of scalability to large numbers of nodes. However, detractors thought this might be a waste of time because commercial companies will make progress in this area faster than the research community.

10. MISCELLANEOUS TOPICS

10.1. Physical Data Base Design

There was a consensus that researchers should build automatic physical data base design tools that would choose a physical schema and then monitor the performance of the schema making changes as necessary. This would include adding and dropping indexes, load balancing arm activity across a substantial number of disk arms, etc. Hence, tuning knobs should be removed from the domain of the data base administrator and manipulated by a system demon. There was widespread support for this idea, and most people thought it was only a moderately hard problem.

10.2. Design Tools

Several participants noted that current data base design tools are merely graphical drawing systems. Moreover, there are a large variety of such products commercially available at the present time. As such, they have little research appeal. A couple of participants expressed interest in enhanced design tools that include methodological concepts for objects, operations and constraints.

10.3. Real Time Data Bases

The needs of applications which must run in real time was discussed at some length. Monitoring applications typically input a continuous stream of data into a data base. Often they discard data once it is no longer timely. Many times there are deadlines to deal with (i.e the DBMS must respond by the time the part goes by the robot arm on the assembly line). The notion of a transaction in this environment is unclear, and the need for triggers is apparent. There was support for research in this area to look at this class of issues.

10.4. Data Models

There was no support for any more data models. The participants universally felt we had enough already. There was also little support for an attempt to standardize on a common "next generation" data model. This was felt to be an exercise in frustration as well as merely the invention of yet another data model. It was, however, pointed out that OODBs and active databases in a way provide their own data model. In contrast to traditional models, these data models will not be fixed but dynamically changeable through the definitional capabilities of the respective data languages.

10.5. Data Translation

The problem of data translation in a heterogeneous computing environment was raised by one participant. This area was discussed and most people believed it to be a solved research problem. The literature of the early 1970s contains appropriate architectures and real world data translation systems have been deployed in the field for quite a number of years.

10.6. Information Exchange Via Data Bases

A couple of participants raised the problem of information exchange and collaborative work. They argued that there are many application areas where the different end-user participants must utilize various tools which should run on a common representation. In CAD, simulation, and document applications, the absence of a common representation is seen as a severe drawback. There was support for information interchange via a data base in these areas in preference to the current techniques as this would provide automatically features like synchronization, concurrency, recovery, versioning, and security. This approach would also naturally lead to more sophisticated data dictionary systems represented through the schemas of those DBMSs.