3rd National Conference
Databases for Science
INFOBAZY 2002

24–26 June 2002
Gdansk-Sobieszewo, Poland

Selected Papers and Abstracts
guest-edited by Antoni Nowakowski

Database Systems for Tomorrow: New Challenges and Research Areas

Krzysztof Goczyła

Department of Applied Informatics, Gdansk University of Technology, Narutowicza 11/12, 80-952 Gdansk, Poland, kris@eti.pg.gda.pl

(Received 30 October 2002)

Abstract: Since the mid-80s, considerable progress has been achieved in relational database technology. The main achievements have been in high performance, high reliability and availability, scalability and development tools. However, the environment for database systems is rapidly changing. There are new challenges that originate from the present hardware technology achievements, as well as from new kinds of data resources that hardly conform to the well-established relational data model (*e.g.* data from the Web). In the paper, we present the new challenges and research areas, as well as motivations behind them.

Keywords: database systems, new architectures, Web technologies, non-relational models, integration

1. Introduction

Contemporary relational database management systems (RDBMS) are mature, sophisticated software systems that are supported by advanced hardware solutions. RDBMSs are commonly used in all real-life areas where computers are present. Database systems market is large – it is estimated that the annual volume exceeds US$ 10 billion and is expected to grow steadily, even in the face of the recession that the world's economy experiences. This huge amount of money engaged in the database systems business has both advantages and disadvantages. One apparent advantage is that the community of database systems users feel comfortable with technical support, stability and their systems maintenance. There is little chance that the software they purchased (usually quite expensive) will not be maintained and periodically upgraded by its vendor and become an expensive but useless gadget instead. Another advantage is that large RDBMS corporations invest enormous funds into research and development, resulting recently in remarkable advances in such database fields as replication and parallelism. On the other hand, big corporations tend to monopolize the database systems market, which results in narrowing down the development trends into several and quite restricted "safe" areas and prohibiting development of new "risky" technologies. A clear example is the way the object-oriented paradigm [1] is being introduced into the database world. However strange it is, contemporary database technology is apparently the only area of information technology where the principles of object orientation were practically not applied. Of course, some elements of the object-oriented paradigm do appear in object-relational database management systems (ORDBMS) [2], however, they considerably diverge from the full object-oriented model. The latter is fully implemented in object-oriented databases management systems (OODBMS) that still remain on the margins of commercial database applications.

It seems, however, that nowadays the database worlds – both the commercial one and the research one – face completely new challenges that will force changes in database technology much deeper than those which occurred at the end of the previous century. In the nearest future, we will face serious changes in computer systems technology for large database systems. At the same time, the unstoppable growth of the Internet and its informational resources accumulated in the ubiquitous World Wide Web create completely new requirements for functionality of data repositories and data analysis tools. In the following sections, we will take a closer look at these causes and prerequisites, referring them to research areas that should be explored in order to cope with these new requirements.

2. The challenges

One may formulate three main reasons for undertaking new research in the database systems technologies:

1. As a result of the rapid development of the Web technology, it has become quite easy and inexpensive to make information of any kind and quality available to millions of potential information "consumers".
2. Many new applications appear that require programs to be integrated with voluminous data of complex and heterogeneous structure.
3. The rapid development of hardware technology invalidates assumptions that various techniques currently applied in database systems are based upon.

2.1. Exploiting information resources of the Web

The Web can be seen as a large, distributed (and unmanaged) database. It would be extremely difficult to estimate its size in bytes (or terabytes), but certainly the number of Web

users (*i.e.* people who have access to workstations connected to the Internet and at least occasionally make use of it) can be counted in hundreds of millions. The number of Internet hosts that run Web servers is of course much lower but, certainly, also large and rapidly growing into tenths of millions. However, the Web is not a "normal", fully functional database – it is not managed in any uniform way. For this reason, the Web is a huge storehouse of data rather than a real database. "Real" DBMSs are present in this storehouse, but play a secondary, back-end role: they are repositories of structured data dynamically used to fill predefined static HTML pages used to present information by Web sites. Although DBMSs are common in such Web services as e-commerce sites or informative sites, still, the majority of information delivered by Web sites is of static nature, with no explicit or implicit structure nor schema.

It is expected that in the near future the proportions will reverse. The majority of information delivered by Web sites will be dynamically created, based on the content of back-end specialised databases. This information will no longer be presented as HTML pages, but rather as XML documents [3], that are better suited for description of data with rich structure. This in turn requires better integration of XML and DBMSs. Nowadays some database systems vendors (IBM, Oracle, Microsoft) offer extensions to their systems that facilitate storing and processing XML data. However, there is no widely accepted standard in this area. Efficiency of XML processing is also not satisfactory, because XML features are implemented as a layer on top of a relational engine.

There is also another facet of dependency between the Web and database systems. Data acquired from the Web, whether created statically or dynamically, must be stored and eventually processed at client systems, so that they can be useful for a group of users they are addressed to (*e.g.* for businesses of some kind). Taking into account the possible size and complexity of data, this creates severe requirements on client tools for management and analysis of Web data. To sum up, users seek powerful and friendly tools for Web data warehousing [4].

2.2. Integration of programs and data

Traditional database systems used to store data only. In contemporary database systems, relational and object-relational, a possibility to store pieces of program code has been introduced. These code pieces take the form of *stored procedures* and *triggers*. Stored procedures may be written in a language native to a DBMS, a language that is a part of an SQL standard [5], or even in a common programming language like C or Java. In the latter case, however, a programmer must tackle with the *impedance mismatch* problem, *i.e.* the problem of data type conversions and transferring database entities into host language variables.

In database systems, a program is still a "second-class citizen": programs cannot be queried, outputs from programs cannot be directly presented as inputs to SQL queries, programs do not have their consistent model. In general, it results from the lack of tight integration between programs and the data they manipulate. Such an integration is postulated in the object-oriented approach to systems development: objects can be treated as modules that encapsulate data and operations. But, as mentioned above, object orientation enters the world of databases rather through the back door, making small and shy steps called "extensions", "cartridges" or "modules" of some kind.

Making the process of integration more active is essential for many big organisations that have at their disposal hundreds, or even thousands, of mission-critical applications that operate in separation, making their interoperability awkward. Such applications could be integrated if they had a common platform – a database that could store them together with the data they consume and produce. Software engineering is another area where the problem of integration is crucial. In this complex information technology area descriptions and specifications of applications being developed (models, dictionaries, interface definitions, *etc.*), their source code and testing data constitute an integral whole and need efficient environments for storing and processing programs and data. Computer aided software engineering (CASE) tools begin to provide such environments and rapid progress in their functionality and power is expected.

2.3. Keeping pace with hardware advances

No one knows where and when the progress in hardware will stop, or even decelerate. Processors become more productive, disks

become larger, communication media become faster. Soon we will be faced with computer systems where the main memory of a terabyte will serve as a buffer pool for databases of petabytes. As a result, practically every table of a relational database will be capable of being kept in the main memory for faster access and data retrieval. This would demand new data structures and access algorithms, as well as new database tools that would accommodate the changed architecture of computer systems.

Database systems equipped with such an advanced hardware will also have to be equipped with advanced, preferably automated, administration techniques and tools. They include automatic installation, configuring, tuning, failure recovery, and even programming. A human will simply not be able to manage such complicated systems due to the volume of information to be interpreted and the extremely fast response needed to react to rapidly changing workloads.

The progress in hardware efficiency will create new demands on scalability and accessibility of DBMSs. In the near future we will need database systems that will efficiently serve a hundred of thousands of concurrent users without trying their patience. Such systems will have to be ready to store and make accessible petabytes of data, processed in parallel by thousands of processors. These figures exceed by two orders of magnitude the parameters of the contemporary commercial database systems. Currently used techniques and technologies do not enable database systems to achieve such dramatic progress.

Another source of workload for database systems are miscellaneous devices and gadgets commonly used in everyday life, like mobile phones, home electronics appliances, magnetic chip ("smart") cards, *etc.* In the nearest future, most of such popular devices will be equipped with processors and the appropriate dedicated software – together referred to as *embedded systems*. We will live in intelligent buildings, drive intelligent cars, shop in intelligent vending machines, cook in intelligent utensils, and so on. Billions of such devices will be able to communicate with their environment via the Internet (or its successor), which will require millions of servers for these "gizmos" to operate. Traditional two- and three-tier software and hardware architectures will certainly not be able to serve small, but numerous applications of this kind. Moreover, the devices will not have a traditional user interface nor any administration interface – they will have to be self-managing, self-configuring, self-tuning, self-identifying, self-protecting, self-repairing, and, eventually, self-destroying.

3. The research areas

In the following sections we present detailed insights into the main research areas following from the above mentioned problems. These areas are expected to dominate the efforts of the database community in the first decade of the 21st century [6].

3.1. Plug and Play *database systems*

The first reason for development of *Plug and Play* DBMSs are computer systems embedded into everyday life appliances, as discussed above. The systems will not be parameterised by an administrator, so they will have to automatically adapt to ever-changing work conditions. The very first step towards this goal is the development of *self-tuning* DBMSs that will be "intelligent" enough to be able to set themselves hundreds of performance parameters that otherwise would have to be set manually (but they could not, for the reasons mentioned in Section 2.3) by database administrators (DBAs). The next step is automatic choice of physical database organisation, for instance automatically changing disk file structures for data storage and indexing. The next, and much more challenging, problem is automatic design of logical database schema with appropriate integrity constraints, followed by automatic binding of these schemas with – also automatically developed – applications like report generators or data presentation tools. One of the measures towards self-tuning is collecting and analysing parameters of workload generated by production environments together with internal system performance parameters. Based on the data collected, the system could choose appropriate values of configuration parameters and system settings, select proper algorithms and data structures, decide on optimisation strategy *etc.*

Another, but equally important, aspect of *Plug and Play* DBMSs is the problem of information discovery. As has been mentioned earlier, the Web is a huge storehouse consisting of heterogeneous information resources. Some of these resources are databases and their share in overall Web information capacity will grow. Future database systems, fully integrated with the Web,

will have to co-operate with other database systems in the Web just after installation in a corporate intranet or in the global Internet. They will have to find the other "friendly" systems, ready to co-operate across the network. There is a clear analogy with discovery by an operating system of new hardware just attached to a computer. To achieve such a high degree of automatic integration, a comprehensive metadata standard is needed that would cover the structure and semantics of all the objects managed by a database system. In other words, databases must have a common language powerful enough to present their schemas and data to the outer world and to make them available to their "friends".

3.2. Large federated database systems

Autonomous databases are called *federated* if they have agreed to conform to a set of rules so that they are able to co-operate on behalf of one application (particularly, of one transaction). Nowadays, the Web may be treated as a huge federated information system, although the rules governing their components are quite loose (in practice they consist of a set of simple protocols like http or ftp). It is expected that in the near future billions of Web clients will make use of millions of co-operating database systems. Such a client (for instance, associated with an embedded system) will not bother about which particular database stores data needed for the device to fulfil its functions. Appropriate data should be reliably delivered by a federated system, which may possibly have to perform a database transaction over thousands of databases. This would pose quite new challenges for query optimisers. First, they must take into account that some of the federated database systems may not be available due to failure or may deny co-operation due to security constraints. Then, data needed for the transaction may be replicated on many database servers, located in various remote places of the whole system. Also, response time for a given query may remarkably depend on the current workload of the network, which traditional optimisers usually neglect. As a result, a cost-based query plan may have to be dynamically modified, reflecting changes in the workload of the federated system.

Another challenging problem in the processing of federated queries is balance between the expected accuracy of query results and the time allowed to perform the query. Let us assume that we need an average salary of a corporation consisting of 1000 departments, each running its own local database. We probably will be satisfied if we promptly receive an approximate value for the average, that can be gradually refined as time elapses. If we are satisfied with the current accuracy, we can stop the execution of the query, or else we can wait for the exact result (if one exists). In a traditional federated system we have to wait for the result of a distributed query until an exact result is produced, whether this accuracy is really needed or not.

The problem of imprecise, approximated data is inherent not only to queries addressed to federated systems, but also in the formulation of the queries themselves. Let us have another exemplary federated query: "Find good Italian restaurants that are located near my home". A system to which the query has been submitted must first refine the concepts of "good" and "near" criteria used in the query. Most probably, the user will not be happy to have to refine the criteria by himself/herself, so the system should be constantly taught by being "fed" with some kind of knowledge concerning how to interpret vague or ambiguous queries. The next problem is how to determine databases that could store data useful for the query. Under consideration there may be thematic databases on restaurants or on tourist places, as well as geographical databases, and others. The problem of integration surfaces once again: a federated system may be composed of databases of different kinds, possibly conforming to different standards and paradigms.

Another crucial aspect of integrating database systems into large federated systems is co-operation between applications running in the federated systems. Let us consider two e-commerce applications: one runs for a manufacturer, the other runs for a warehouse. It is important that the two applications could understand each other so that they start co-operation as soon as they have found each other. To this aim a common language is needed that could be used on the interfaces of the applications running in the federated system or in the Web. This language should be flexible and powerful enough for the applications to communicate their functions and to define their data. A step towards this goal is the *Unified Modelling Language* (UML) [7], primarily devised

for modelling and developing applications rather than precise functional specifications. Together with XML for metadata definition, UML may be a promising tool for exchange of functions and data between applications.

3.3. New database system architectures

The progress in hardware technology, described in the previous section, enables developers to create more and more powerful database applications. The best performance is achieved when applications are run within parallel systems of a *shared-nothing* architecture, where each processor has its own main memory and own disk memory, and the only shared resource is a communication medium. We expect further intensive development of parallel systems that could co-operate in large high-performance and high-availability clusters. It is a great challenge for DBMS software developers. The tasks that have to be considered are: load balancing among parallel system nodes, developing partitioning and replication strategies, and creating optimal query plans for such complex working environments.

As has been pointed out before, soon database engines will be able to use main memory buffers of terabyte capacity. Relational tables crucial for a given application, together with the corresponding indices, will reside entirely in the main memory for a long time. It means, for instance, that traditional access methods based on B-trees will no longer be adequate. Indeed, B-trees are not the best index structures for in-memory access, as they are based on dividing data into chunks (disk pages), which is inappropriate for in-memory data. Other database mechanisms that require re-thinking are transaction handling, recovery, concurrency and all the other techniques that make use of main memory buffers.

The development of disk storage technology manifests itself in rapid growth of disk capacities and transfer rates. As a result, the seek time (*i.e.* the time needed to move disk arms to a disk cylinder needed) becomes the bottleneck of disk storage throughput. This requires development of new storage organisations and new access strategies (*e.g.* access requests scheduling) that would minimize disk arms activity.

Many database applications, in particular in the fields that much rely on visualisation techniques, require large volumes of data. Soon these volumes will attain hundreds of petabytes. This will come to reality when disks have achieved at a reasonable price so large capacities that manipulating such voluminous data will be feasible in the traditional 2-level storage architecture. Another possibility is the introduction of new storage media (*e.g.* based on holography) that would be mature enough to be applied in commercially available computer systems. These new media could be used as a third, the slowest but the most capacious, level of database storage. It is clear that the advent of 3-level storage of exabyte capacity offers novel capabilities in archiving and replication methods, thus increasing reliability and availability of database systems. This can lead directly to "never-fail" systems that, from the user's perspective, are always up and running and never lose data.

In this context, it is worth mentioning the steady interest of vendors and developers in 3-tier architectures of database applications, where only one program (DBMS) runs on the server and only one program (application server) runs in the middle tier. Both programs will have to be capable of serving thousands of concurrent client connections. The problem of scalability of such architectures is a serious challenge for researchers and developers in the information technology industry.

3.4. Unifying applications and data in databases

There are several important aspects of the necessity to uniformly handle data and applications in database systems. Firstly, we need uniform, universal and flexible application models similar to data models used in software engineering and databases. One possibility is to describe each application as a set of business rules and their flows. The flows can be formulated and visualised in the form of flow diagrams, as we formulate and visualise relational data as tables. Presently, there are some systems that support workflows. We can imagine that data from these systems are interpreted and compiled into database triggers to be defined in an active database schema. Executing triggers by a database engine is much more efficient than executing them outside the system, so we can expect considerable gains in performance and flexibility of applications. However, to this end, we need to define thousands of triggers in one database, which for now is not feasible. It is estimated that scalability of three orders of magnitude is required, but

job appears to be worth the while: for each data item we could define a trigger that would execute a (probably tiny) action for an application running over a federated system. Let us imagine for instance a stock application that promptly informs stakeholders of any change in the quotation of a given set of shares.

Secondly, it is of crucial importance for software engineering that the component-based approach for application development should be integrated with databases. It would be desirable if we could build any database application (maybe not only a database application) from ready-to-use components stored in a database. Obviously, such applications would be easily and efficiently stored and executed on demand within a database system. Presently, there is no widely accepted uniform and consistent model of software components. Different vendors promote their own component systems: CORBA, OLE, COM, DCOM, EJB, JINI *etc.*, so maintaining them in one database, although theoretically possible, does not seem neither reasonable nor comfortable for use. One, but far from satisfactory, currently available solution is the possibility of defining *User-Defined Types* (UDT), postulated in the SQL-99 standard. Actually, UDTs correspond to classes in the object-oriented paradigm. However, the standard does not treat UDTs as active components for different applications, but rather as mere types of data to be stored in a database. Another possibility stemming from the SQL-99 standard is coding software components in a database procedural language like SQL/PSM [5], primarily aimed at coding stored procedures. It seems that further development and refinement of languages of this class could be fruitful, particularly if it is accompanied by efforts towards optimisation of execution, similar to the optimisation of SQL queries.

One consequence of realizing a component-based model of database application development will be a need for new application development tools. The tools should be able to help a user to find necessary components, to integrate them into an application, to test the result in a testing environment and finally to move it into a production environment. It seems that the first step – finding an appropriate component – is the most difficult one, as it requires a user to specify his/her needs. This in turn requires a standard, precise component specification language, unless we want to make a user browse through numerous descriptions of components that might be useful (but almost always are not). We hope that such environments will appear together with more mature and advanced object-relational systems.

3.5. Integrating structured and semi-structured data

Traditional database systems, as well as the "next generation" systems [2], store data that have a well-defined structure known a priori, called a database schema. Data of this kind are *structured data*. The uncontrolled growth of the Web causes that more and more information resources (useful anyway) contain data that are irregular, incomplete or of complexity that can hardly be described by relational or even much richer object-oriented data models [8]. Data of this kind are called semi-structured data. A convenient tool for the description of such data is *eXtensible Markup Language* (XML), that allows for alternatives, optional constructs, multivalued attributes and other means that go beyond classical database constructs. As a consequence, contemporary advanced database systems are capable of storing XML data and querying them in a way similar to querying structured data (*i.e.* using a declarative query language).

It is expected that the Web content coded in XML will soon prevail the content produced in HTML. This will allow for automatic analysis of Web pages, including extraction of semantics from Web data, which is hardly attainable in the case of unstructured, HTML data. Additionally, XML documents can be self-descriptive in the sense that they may be accompanied by metadata formulated as *Data Type Definitions* (DTD) that actually play the role of database schemas. The *Document Object Model* (DOM) [9] attemps to standardize the form of XML published documents.

Presently, commercial DBMSs handle XML data via middle layers that translate them into a relational form. There is, however, no standard for processing XML data, although XPath [10] language is a good step in this direction. Actually, there is no clear vision of how to integrate XML Web resources and database technology. There are also a lot of important problems to be attacked in this area, like efficient processing of deeply nested hierarchical objects,

typical for XML documents, the development of appropriate transaction models, devising new access methods (including methods of updating XML data), versioning and configuration management.

4. Conclusions

The amount of information we have at hand grows exponentially. The information is of varying quality and structure – from highly structured, "clean" data, appropriate for controlling devices or managing big enterprises, to irregular, imprecise and inconsistent data, distributed via Web sites of varied origin. Such situation creates new challenges for database systems, whose main task has always been to organize, store and make available data in a way most adequate for a given application. The main challenge can be formulated as follows: The priority is to develop database systems that would be able to collect, organize, store, analyse and make available all information resources of humankind in such a way that the information could be used on-line by anyone.

It is clear that this general goal is strongly related to Web technologies. Firstly, in the nearest future, most of our knowledge will be digitised (in the form of *digital libraries*) and made available globally through the Web. Secondly, the number of Internet users grows so fast that very soon most of the Earth's citizens (at least in the more developed areas) will become consumers of this knowledge, and consequently, clients of the knowledge repositories. Many of them will also become producers of knowledge, which will also require access to knowledge repositories. An ideal database system that we aim at should be able not only to respond to any queries formulated by any user of the global network, but also to anticipate users' queries and actively present useful information. As a matter of fact, we should strive to transform the huge storehouse of data called the Web into an integrated, intelligent, global information system based on an advanced and mature database technology.

References

[1] Cattell R G G and Barry D K (Eds) 2000 *The Object Data Standard: ODMG 3.0*, Morgan Kaufmann Pub.

[2] Stonebraker M and Brown P 1999 *Object-Relational DBMSs: Tracking the Next Great Wave*, Morgan Kaufmann Pub.

[3] *Extensible Markup Language (XML) 1.0*, Second Edition, W3C Recommendation, October 2000, www.w3.org

[4] Hackathorn R D 1999 *Web Farming for the Data Warehouse*, Morgan Kaufmann Pub. Inc.

[5] Gulutzan P and Pelzer T 1999 *SQL-99 Complete, Really*, R&D Books

[6] Bernstein P, Brodie M, Ceri S, De Witt D, Franklin M, Garcia-Molina H, Gray J, Held J, Hellerstein J, Jagadish H V, Lesk M, Maier D, Naughton J, Pirahesh H, Stonebraker M and Ullman J 2000 *The Asilomar Report on Database Research*, www.acm.org/sigmod/record

[7] *OMG Unified Modeling Language Specification, Version 1.4*, September 2001, www.omg.org

[8] Florescu D, Levy A and Mendelzon A 1998 *ACM SIGMOD Record* **27** (3) 59

[9] *Document Object Model (DOM) Level 2 Specifications*, November 2000, www.w3.org

[10] *XML Path Language (XPath) Version 1.0*, W3C Recommendation, November 1999, www.w3.org

Provision of Databases in the Poznan Supercomputing and Networking Center

Sławomir Niwiński[1], Iwona Pujanek[2] and Maciej Stroiński[1]

[1] *Poznan Supercomputing and Networking Center, Noskowskiego 10, 61-704 Poznan, Poland, {niwinski,stroins}@man.poznan.pl*

[2] *Poznan University of Technology, Main Library, Plac Skłodowskiej-Curie 5, 60-965 Poznan, Poland, iwona@ml.put.poznan.pl*

(Received 16 October 2002; revised manuscript received 13 January 2003)

Abstract: The article presents a concise report on the experience gained in the last three years in the scope of network provision of bibliographic databases by the Institute for Scientific Information (ISI) and full-text humanities and medical databases by EBSCO Publishing. The authors emphasise the importance and impact of the programme and the databases, co-financed by the State Committee for Scientific Research, on the initiation and continuation of organisational activities and efficient database access management. The paper contains a short review of the information resources presently available, including the titles of bibliographic and full-text databases, the scope of licences, subscription periods, and the volumes of archival resources. It provides statistics illustrating the distribution and extent of the bibliographic database usage by the scientific community, including, active institutional and individual, users and discusses the hardware and software used to provide the network database access services, the availability conditions, as well as the rules of license renewal and co-financing by the interested institutions. The report also deals with the access conditions and access abilities to the electronic versions of humanities, economics and medical databases offered by EBSCO Publishing. It is vital to show the