Bulletin of the Technical Committee on

Data Engineering

December 2004 Vol. 27 No. 4

IEEE Computer Society

Letters

Special Issue on Database Support for the Sciences

Conference and Journal Notices

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems.

There are two Data Engineering Bulletin web sites: http://www.research.microsoft.com/research/db/debull and http://sites.computer.org/debull/.
The TC on Data Engineering web page is http://www.ccs.neu.edu/groups/IEEE/tcde/index.html.

Letter from the Editor-in-Chief

The Current Issue

The September issue of the Bulletin was on dealing with biological sequence data. That issue described some of the work going on to make database systems a suitable tool in the biology area. The current issue is on the broader issue of how database systems can be relevant to many of our sister sciences, including but not limited to biology. The traditional target for database technology was business data processing. Scientific applications present a new set of requirements with the potential to push database technology in new directions. What is interesting, however, is how successful one can be by using existing database systems, and by extending them in directions that the database community was already inclined to pursue.

Scientific applications are tremendously important. But they represent only a modest business opportunity in terms of market size. So the fact that science can be well served by existing technology or extensions to this technology is a boon, both to database researchers and to the scientists in other disciplines that are increasingly served by database technology. Thus, what other disciplines seem to need are more capability to manage metadata, additional search strategies and techniques, additional scalability (to the petabyte range), improved data visualization, automated data translation, and better tools, in some cases for very mundane activities like loading data. These are capabilities that database systems researchers have a long interest in and are continuing to improve. So the moral of this story is that databases can improve in their ability to serve their traditional market, which after all also needs these features, while they serve the needs of the scientific community.

Meral Özsoyoğlu, the editor of the current issue, has brought together an enlightening collection of papers from database folks who are collaborating with scientists from an interesting subset of our sister sciences, biology, astronomy, engineering, etc. The emphasis here is on database technology, both how it can be extended, and how it can be adapted to the needs of scientists in these disciplines. The issue is not on the science of the other disciplines, except as that is needed to describe the motivation for extending and adapting. As is characteristic of Bulletin editors, Meral has reached out to both academic and industrial database specialists in putting together the issue. This always requires of an editor determination, patience, and hard work. I want to thank Meral for her very successful efforts on this fine issue, which should be of interest to a very large segment of the database community.

David Lomet
Microsoft Corporation

Letter from the Special Issue Editor

As the sciences are becoming more and more data rich, most scientists are facing the challenges of dealing with ever increasing volumes of data in their research. It is generally agreed that tools are needed for modeling and organizing large amounts of scientific data for efficient access and easy querying that will work for domain scientists. While the database and data mining tools/techniques seem to be a perfect match for managing and analyzing scientific data, building such tools to help scientists requires working collaborations between domain scientists and computer scientists, more specifically database and data mining researchers, and involves several challenges. This issue of Data Engineering Bulletin explores the potential and challenges for database research to help other sciences in this data-rich era, and presents a spectrum of database techniques and approaches for scientific applications.

In the first paper, Jim Gray and Alex Szalay discuss the potential and the need for ”Bridging the Gap between Databases and Science” in general. They also discuss what is needed for making such a collaborative research be successful based on their experiences in building the World-Wide Telescope and the Sky Server Project. As more powerful telescopes are being built, more data is being generated, and the volume of data to be processed in Astronomy is reaching challenging proportions. The next paper also describes a collaborative research by database and astronomy researchers. Calimlin, Cordes, Demers, Deneva, Gehrke, Kifer, Riedewald, and Shanmugasundaram describe their initial approach and the challenges in building a data management and data mining system for astronomy data for the Arecibo Telescope.

Data volume and workload requirements make scientific databases as good candidates for automated physical design techniques. Papadomanolakis and Alimaki discuss database partitioning techniques by utilizing the workload information, and present an automatic schema design tool for partitioning database tables. They evaluate this approach experimentally using the astronomy data from the Sky Server project. Howe and Maier advocate in situ processing of large scientific data repositories as a means of harnessing database techniques. They argue that using a data model interface by superimposing a specialized data model over native data repositories can provide the benefits of using a database management system without the substantial cost of massive bulk loading and reloading that come with the database deployment.

In many scientific database applications, similarity search is either commonly used by itself, or as a building block (e.g., clustering, phylogeny construction). Kriegel, Pfeifle and Schonauer discuss similarity search, and the utilization of a set-based similarity measure in two diverse applications in science and engineering domains. Data provenance (or data lineage) is another very important problem in scientific databases. Supporting data provenance in bio-scientific database applications is becoming more important with the explosion of curated databases in molecular biology and genomics. However, there is relatively few previous work on tracing the lineage of data in scientific databases. Wang-Chiew Tan, in her paper, gives motivation and an overview of research in this important subject, and presents a prototype for tracing the provenance and flow of relational data.

Graphs are very useful in modeling data in scientific problems and applications. The last two papers of the issue are on graph databases. M. Ozsoyoglu, Nadeau, G. Ozsoyoglu, and Tasan present an integrated set of tools for querying, browsing, and visualizing metabolic pathways in multiple abstraction levels, where pathways are conceptually modeled as a graph structured database. Finally, Stephens, Rung and Lopez briefly describe support for graph data in a commercial database management system.

I hope you will enjoy this issue, and find it useful and interesting. My special thanks to all the authors for their contributions, and Wanhong Xu and Murat Tasan for their assistance in the assembly of this special issue of the Bulletin.

Z. Meral Özsoyoğlu
Case Western Reserve University
Cleveland, OH, USA

Where the Rubber Meets the Sky: Bridging the Gap between Databases and Science

Jim Gray
Microsoft Research
Gray@Microsoft.com

Alex Szalay
Johns Hopkins University
Szalay@pha.jhu.edu

Abstract

Scientists in all domains face a data avalanche - both from better instruments and from improved simulations. We believe that computer science tools and computer scientists are in a position to help all the sciences by building tools and developing techniques to manage, analyze, and visualize peta-scale scientific information. This article summarizes our experiences over the last seven years trying to bridge the gap between database technology and the needs of the astronomy community in building the World-Wide Telescope.

1 Introduction

If you are reading this you are probably a "database person", and have wondered why our "stuff" is widely used to manage information in commerce and government but seems to not be used by our colleagues in the sciences. In particular our physics, chemistry, biology, geology, and oceanography colleagues often tell us: "I tried to use databases in my project, but they were just to [slow | hard-to-use | expensive | complex]. So, I use files." Indeed, even our computer science colleagues typically manage their experimental data without using database tools.

What's wrong with our database tools? What are we doing wrong? We have been trying to answer those two questions in the context of astronomy data - that is, we have been working for the last seven years to try to bridge the gap between database products and the community of professional astronomers. In the process we have come to realize that science is being transformed - most sciences are becoming data-rich. To deal with the data avalanche, each discipline is growing a data analysis arm that collects large datasets representing mankind's knowledge of the field. These datasets are augmented with applications that encapsulated mankind's understanding of the data and our understanding of that branch of science. Each field is being objectified - it is building an object model that represents the facts and building methods that allow easy selection and recombination of the facts. This represents one of the most exciting arms of these sciences - and database and data mining technologies are the heart of these developments.

Put more succinctly: each science was empirical at first and then grew an analytic branch. Recently the analytic branches have spawned computational branches that simulate the analytic models. So the X department of most universities has a computational-X faculty (e.g.: computational-linguistics, computational-genomics, etc.). These computational scientists are adding to the data avalanche. Now we are seeing the emergence of X-info departments to deal with the avalanche (e.g., bio-informatics). The X-info community is trying to put all

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

the data and literature on the public internet, federate it, and provide tools that make it easy for domain scientists to find information, and make it easy for scientists to add their data and findings to the corpus. This will redefine the notion of "publishing" and has deep implications for how the world's scientific enterprise is organized.

The vision is *eScience* in which the unified experimental, theoretical, and simulation data and literature are at your fingertips. You can explore all the world's scientific data and literature looking for patterns and anomalies. Tools that encapsulate the best statistical algorithms, the best machine learning algorithms and the best data visualization are at your disposal to make you more productive and a better scientist. When you want to publish your findings, all your work has been recorded, so that others can follow your arguments and extend your work. That's the vision; but, the reality is far from that vision.

Historically, scientists gathered and analyzed their own data. But technology has created functional specialization where some scientists gather or generate data, and others analyze it. Technology allows us to easily capture vast amounts of empirical data and to generate vast amounts of simulated data. Technology also allows us to store these bytes almost indefinitely. But there are few tools to organize scientific data for easy access and query, few tools to curate the data, and few tools to federate science archives.

Domain scientists, notably NCBI (ncbi.nih.gov) and the World Wide Telescope (WWT, ivoa.net), are making heroic efforts to address these information management problems. But it is a generic problem that cuts across all scientific disciplines. A coordinated effort by the computer science community to build generic tools could help all the sciences. Our current database products are a start, but much more is needed.

We have been participating directly in the WWT effort. This is a personal, anecdotal account of what we learned working with astronomers and advice to others on doing similar things for astronomy or other disciplines.

2 Science is Changing

For each science discipline X, X-info and comp-X are controversial. There is a long-tradition of using primary sources – gathering and analyzing your own data rather than using second-hand data or synthetic data. The traditional primary-data approach is still valid, but scientists increasingly need to compare their data to large shared databases (e.g. is this DNA sequence already in Genbank? If so, what is known about it?) In some cases advances come from analyzing existing data sources in new ways - the pentaquark was found in the archives once theoreticians told us what to look for.

Traditionally, data analysis has followed an *ftp-grep model*. In the ftp-grep model, the scientist first gets a copy of the relevant data from an archive: that is she uses ftp (file transfer protocol) to copy the relevant data sets from a science archive on the internet to a local computational server. The next step is that the scientist uses an analysis package to scan through the data looking for patterns. We use the name grep (generalized regular expressions) to describe this process. In fact the analysis program is often a "dusty deck" written long ago in Fortran invoked with a scripting language like Python.

The ftp-grep data analysis approach is breaking down as datasets grow to terabyte and petabyte scale. First, although you can ftp a megabyte in a second, it takes days or years to ftp a terabyte to your local server - and most local servers do not have spare terabytes of space. But datasets are growing to the petabyte range. So, scientists must to learn to move the questions-and-answers across the internet, and do the data analysis near the data whenever possible.

Section 4 explores this data-archive and portal architecture in more depth. But here, we want to make the point that the skills and the workflow of next generation science will be different. The new breed will need new tools and need the skills to use them.

Traditionally science data was included in tables and graphs that were part of the science article. This metaphor is breaking down or has already broken down - the data is just too large to be printed. Rather the data is placed in a project website (like BaBar) or is deposited with a science archive like Genbank or Astrophysics Data System. So, the concept of publishing has morphed so that the "paper" is about three things:

(1) “how to read the data”, (2) “here is what we found”, and (3) “here are some things we are going to look for next in the data and things you might want to look at.”

In this new world, data is published on the Internet and the literature mostly summarizes it, is a guide to it, and presents some interesting conclusions. Not all science is like this, but the trend is definitely increasing. Even more interesting is the fact that new results come from new combinations of existing data. It appears that there is a “Metcalfe’s law” for datasets: the utility of N independent data sets seems to increase super-linearly. One can make $N(N-1) \sim N^2$ comparisons among the datasets and so the utility is approximately N^2.

The increased utility of being able to combine multiple independent datasets is one of the drivers for the digital libraries being built by the various science disciplines. The biology community has impressive portals in Entrez, DDJB, and Esembl - they integrate all the data and the literature. They are adding new datasets rapidly. They represent an architecture where one group manages both the data and literature all in one location. An alterative approach, typified by SkyQuery.Net lets the data reside at the data sources and a web portal (SkyQuery.Net in this case) federates them via web services.

We discuss these architectural alternatives in Section 4. The point here is that the World Wide Digital Library for each discipline X is quickly evolving - mostly driven by the domain scientists. With time we expect these many libraries to grow together to form one giant library. The architecture for these federated libraries and for the federation of libraries stands as a challenge for the computer science field. If we fail in our responsibility to address that challenge, the domain scientists will certainly come up with solutions and we computer scientists will have missed an opportunity to do something really valuable.

One problem the large science experiments face is that software is an out-of-control expense. They budget 25% or so for software and end up paying a lot more. The extra software costs are often hidden in other parts of the project - the instrument control system software may be hidden in the instrument budget. The data pipeline software budget may be hidden in some of the science budgets. But there is no easy way to hide the cost of the archive in some other budget. So the archive software often does not get written at all. In this scenario, the data is deposited in a poorly documented ftp-server and is never connected to an easy public access.

Funding agencies want scientists to publish their data, but cannot afford to pay for the data archives. One thing we computer scientists could do is to make it much easier and less expensive to publish scientific data.

3 Rules of Engagement - Working With X

How can you, a computer scientist, engage with a domain scientist (or group)? You might try going native - switch from computer science to bio or eco or geo or X science - stop reading all those old CS journals and give up going to all those computer science conferences.

Certainly, we have seen going native “work” but it has two shortcomings (1) you are giving up a lot, (2) you are unlikely to be able to help domains Y and Z who need you just as much as X. So, our advice is to stick to your expertise. Keep your office, your CS appointment and your magazine subscriptions; but, start a collaboration with some scientists and over time broaden that collaboration to work cross multiple disciplines.

So, how do you engage? Well, first you have to find someone in domain X who is desperate for help. Frankly, most scientists are quite happy with the status quo. They are collecting their data; they are publishing their papers; and they are generally working the way they and their colleagues have worked for decades if not centuries. All this stuff about a data avalanche is not really their problem. There is however a small group of pioneers in most fields who are absolutely desperate for help. They are pioneering a new instrument, or have a new simulator that is indeed producing the data avalanche. The answer is somewhere inside the terabytes if they could only get the data processed, organized, and then be able to query it in finite time.

You will recognize these people when you meet them - they are the ones with the jobs that take weeks or months to run their Python scripts. Their delay from question to answer is days or weeks. They are the ones who are doing batch processing on their data. They envy people who have interactive access to data and envy people who can explore their data. They are the ones who are desperate for help. Desperate enough to even spend time on a long-shot like working with you.

OK, so suppose you locate a desperate person. How can you help? How can you communicate? Well, first you have to learn a bit of their language. This generally involves mastering the introductory text for that domain - it is painful and you can skip this step if you are in a hurry, but you will end up doing this work in any case. Doing it early is the most efficient way. In parallel you have to form a working relationship with the domain experts (scientists.) You need to put in enough face time so that they are not surprised to see you. This goes hand-in hand with developing a common language. The converse of this, the domain scientists you are working with need to explore some of the things done in computer science and in other disciplines so that they have a sense of what is possible and what is almost possible.

Once you have established communication, you need to find problems that leverage your skills. This requires you do some ethnography of Domain X. Do a time-and-motion study of how the scientists are currently working, and look for some low-hanging-fruit; simple tools you could build or deploy that would be a major productivity boost. You are looking for a project that may take you weeks or months, but that would save them years.

To give a specific example, in the context of the Sloan Digital Sky Survey data system, we developed a set of 20 questions each of which would have taken a week or more of programming and a week of execution with the then-existing tools. We also developed 12 visualization tasks (scenarios) that typify the kind of data visualization that scientists want to do with the Sloan Digital Sky Survey data[Szalay00].

These 20 questions then became the requirements document for the design of the SDSS Archive server. It had to be easy (less than an hour) to ask each of these questions and the answers had to come back within minutes. This was a contract that both sides (astronomers and computer scientists) could understand.

Since that time, we have seen this 20-questions approach applied to many other branches in astronomy. Typically the new group adopts about half of the 20 questions from the original set but discards about half and adds ten more that are unique to that branch or instrument.

Our rough observation is that scientists spend a lot of time loading, scrubbing, and conditioning data. Once they have the data in acceptable form, they spend their time looking for haystacks and for needles in haystacks - that is, looking for global properties and global trends and patterns (the haystacks), or looking for anomalies (needles in the haystacks). Computer scientists can help automate the data handling processes, they can help with the data analysis, and they can help with the scientists ask questions and understand answers.

It seems computer scientists can be roughly divided into three groups: (1) Data visualization experts who are good at capturing user's questions and good at visualizing answers in novel ways. (2) Data analysis experts who are skilled at statistical analysis, data mining algorithms, and machine learning algorithms, or just algorithms in general. (3) Software systems people who are good at data management or systems management or workflow management. The domain scientists will likely not understand these broad distinctions, but computer scientists know that these three communities are largely segregated - having their own conference and journals.

Given these observations, it seems obvious that the Visualization and Analysis computer scientists can help improve existing systems, but that the logical progression is for the systems people to build a "rationalized base" and then for the Algorithms and Visualization people to build new tools atop that base. So, we have been working away to build both individual astronomy archives and to federate that as described in the next sections. We are certainly not done building the base, but we think we are ready to accommodate any Analysis or Visualization experts who want to apply their techniques to this corpus.

A rough chronology of our collaboration over the last seven years is that the twenty queries suggested a logical and physical database design - and also showed the need for a spatial data access method. About 25% of the queries involved spatial clustering of objects. Addressing these issues took a year or two and naturally led to an "outreach website (SkyServer.SDSS.org) that provides both public access to the data and is also the basis for online astronomy education for pre-college students. There are now 150 hours of online astronomy instruction in several languages that teach both astronomy and teach tools and skills to analyze a large online scientific database. This education component comprises more than 10% of the traffic on the SkyServer website.

At this point, data loading had become an onerous task - the data kept coming and being reprocessed. So, we automated the loading process with a workflow system built atop SQLserver and its Data Transformation Service (DTS.) With the SkyServer in place we began exploring architectures for the World Wide Telescope (the Virtual Observatory) federating all the world's astronomy archives. A pixel server (image cutout) web-service was the first experiment with web services. It led to the SkyQuery (SkyQuery.Net) architecture.

In parallel, several other groups did their own 20-questions exercise and cloned the basic SkyServer database design and spatial access methods for their archives. Recently we have been refining and extending our spatial access techniques, redoing our design in light of the revolution in database architectures (the unification of data and methods), and also preparing for the next-generation surveys which will capture a thousand times more data.

In summary, it has been a very productive collaboration. It has in part answered the question: "Why don't they use our stuff?" The answer is indeed that databases are hard to use. Paradoxically, it is difficult to get data into them - data scrubbing and loading is labor intensive. Organizing data is challenging. Once the data is loaded, cognoscenti can do wonders, but "normals" do not have the data analysis skills to pose queries and to visualize answers. As the community evolves, and as some scientists master these skills, they communicate successful patterns to their colleagues - that is how scientific techniques evolve. It is an organic process.

In closing this section on "how to collaborate" we confess that we are very data centric. Computer scientists might contribute to domain X by bringing new ways to think about dynamic systems or new notations for dynamic systems. Differential equations are the standard way to describe dynamics 00 but they are not particularly good at describing discrete event systems. Computer scientists have struggled with and modeled discrete event system with tools like simulation languages, process algebras, and temporal logics. It is quite possible that these new ways of describing discrete events could provide breakthroughs in process-oriented sciences as diverse as ecology, metabolic-pathways, or social interactions.

4 Architecture: Services and Portals.

The broad architecture of the World Wide Telescope and of other digital libraries has emerged over the last decade. They will be federations of data servers, each providing access to its data resources. The federation will be unified by one or more portals, typically collocated with one of the mega-servers. Each portal provides high-level tools to access the "local" archive and also to correlate that data with the other data sources in the internet and with the scientific literature.

This pattern is emerging in other disciplines - for example Entrez, DDJB, and Esembl show the same pattern in the bio-informatics domain. A mega-archive co-located with a portal providing tools that integrate and federate that archive with the other important archives.

The Service-Portal architecture requires that data from different services be interchangeable - that there be standard terms and concepts across all archives. This requires *objectification* of the discipline - an object model that encapsulates and represents facts in standard ways. The astronomy community has embraced the UCD methodology as a core vocabulary for units and facts[UCD]. From a database perspective - this is the schema.

Built atop this, the astronomers have defined a few web services that, given a query, return the answer. These services are defined in terms of XML Schema and SOAP messages. The first service returned all objects within a circle (cone)[Cone]; the second returned an image of an area of the sky[SIAP]. There is also a spectrogram service. A dozen generic services have been defined so far[VO-services].

The portals build on top of these simple services providing ways to combine data from multiple archives. This is nascent now - there are only a few portals, but each of them is evolving rapidly.

One unanticipated benefit of developing the SkyServer architecture and placing it in the public domain is that it has been copied about ten times by other archives. The server has a spine schema consisting of the spatial data organization and the general organization of optical and spectroscopic data. Other surveys have discarded the SDSS-specific attributes and replaced them with their own. The result is that the Royal Observatory of Edinburgh, Space Telescope Institute, Cornell, Caltech, and NOAO have each been able to leverage this software base. In general, a survey can be converted to the spine schema in a matter of hours if it is small (gigabytes) or a matter of days if it is large (terabytes).

The common spine schema and web services interface has also enabled us and others to add many datasets to the SkyQuery interface. Once the data is online, the web services can be defined and it can be registered as a member of the SkyQuery federation in a few hours[VO Registry].

The portal itself has evolved in two un-anticipated ways: (1) users can define personal databases that represent their "workspace" on the server, and (2) users can submit long-running queries to a batch system that schedules these jobs and deposits the answers into the user's personal database. Both these features are described in [Nieto-Santisteban 04b] and are briefly sketched here.

Users routinely perform data analysis in steps - using the results of previous steps in subsequent analysis. The often start the analysis with datasets that they have developed over the years. To facilitate this, the portal lets a user define a database that is private to that user. The user can create tables in that database and upload private data to it. The user can deposit query answers to existing or new tables in this database, and can use tables from this database in subsequent queries. Access control is based on username and password (better security mechanisms are being discussed). The database owner can grant read or write authority to the personal database. This is convenient for collaborators who share a workspace and is a convenient way to publish data to your colleagues. The databases are currently limited to 1GB, and so far no one has asked for more - but we see no reason to limit the databases to such a small size in the future.

The portal has limits on how much processing a query can consume (elapsed time) and how large the outputs can be. For the public site the limits are 90 seconds elapsed and 1,000 rows of output. For the collaboration site the limits are 90 minutes elapsed and 500,000 rows of output. Both categories occasionally need to exceed these limits, and it is often the case that the user does not want to wait for the job to finish. This obviously suggests a batch job scheduler. Users can submit jobs to this scheduler which then runs them in the background. If a job exceeds its quota, there is an option to double the quota and re-run the job (up to some limit). There is also a simple web interface to check on the status of jobs. The easiest way to understand this system is to just try it: create a personal database and run some sample queries[CAS].

5 Was It Worth the Effort?

After all this work, what are the benefits? Are the scientists better able to access their data? What breakthroughs has this enabled? At best, we can answer these questions with anecdotal evidence.

The SkyServer website has served about 60M web hits. More than 10% of that traffic is to the educational part of the site - so the "outreach" has been very successful. The collaboration SkyServer website is currently processing more than 100,000 SQL queries per month, the public site processes many more and the CAS Job system *is* processing nearly 10,000 jobs per month. So, the system is being used. The astronomers have wrapped the web services in Emacs and Python scripts. Many use them as a matter of course in their work.

There are two side-by-side comparisons of the files-vs-database approach. One compared a "Grid-Condor" design for finding galaxy clusters to a database-index-parallelism approach. The DB did the job better, simpler, and 50x faster[Nieto-Santisteban 04a]. In a second experiment, a colleague had written a program to find near-earth objects (asteroids). The program was about 20 pages of code and ran for three days. We wrote a 50-line SQL program in a few hours that got the same answer in about 30 minutes. When we added an index to support the query, it ran in 3 minutes. When questions and answers can be done in minutes it changes the kinds of questions you ask - this seems to us to be an example of the breakthrough we are talking about.

The SDSS is a side-by-side optical and spectroscopic survey. Spectrograms give precise redshifts and hence approximate distances based on Hubble's constant. We use object spectra as a "training set" to estimate even more galaxy distances based on photometry alone - the so called *photometric redshifts*. We found that one category of objects was significantly under-sampled by the spectroscopic survey and so our photometric redshift estimator was less sensitive than it might be. The code to schedule and "drill" the fiber plate to take spectra was huge and poorly understood - so the astronomers could not change it easily to address this custom problem. In an evening we wrote the queries to select a set of observations and generated a drilling list (holes in an observing plate). The observation was scheduled and a few months later we got the observation data. Within hours the data was loaded into the database and the redshift estimator was recalibrated with the larger dynamic range and substantially improved accuracy.

Our colleagues have much better access to SDSS data than they had to any previous survey. Not just that, having a database has improved data quality. Foreign keys, by having integrity constraints and by doing integrity checks in the data loading process have uncovered myriad problems which have fed back into the pipeline.

The SkyServer embeds the data documentation in the DDL (using the UCD notation and terms[UCD].) A tool scans this documentation and automatically generates online documentation for the website. So, once the astronomer has defined the DDL, he can use the automated loading tools (DTS) to load the database, and the website comes up with access to the data and with online documentation - immediately. This has been a boon to several projects - saving months if not years of development.

In summary, we think that astronomers are finding that this is the easiest way to publish their observational data. It could be better, but it is a substantial advance over what existed. Now the challenge is to make the process even more automatic, deal 1,000x more data volumes, and provide the missing visualization and analysis tools.

6 Some Lessons Learned From the WWT

Astronomy is a good example of the data avalanche. It is becoming a data-rich science. The computational-Astronomers are riding the Moore's Law curve, producing larger and larger datasets each year. The observational astronomers are also riding Moore's law, deploying denser CCDs and more instruments each year. The total data volume seems to be doubling every year. We have learned some interesting lessons from managing the Sloan Digital Sky Survey dataset. It is about 40 TB of raw image data which places it somewhere between Genbank (50GB) and BaBar (1PB) in the spectrum of scientific datasets.

Data Inflation Paradox: You might guess that derived data, the processed astronomy data would be 10% of the size of the raw instrument data. After all, the night sky is mostly black. Your intuition is approximately correct. The *Data Inflation Paradox* is that after a few years the derived data products derived data are most of the volume. NASA has a useful classification for data products: Level 0 is the raw instrument data, Level 1 is the calibrated data, and Level 2 are derived data products built from Level 1 datasets. As time passes, the Level 0 data arrives at an approximately constant rate. But, a growing number of Level 1 and higher data products are derived from the raw data - star catalogs, QSO catalogs, cluster catalogs, materialized views that are particularly convenient. The derived products are indeed much smaller - but they are also much more numerous.

Versions are a second data inflation driver. The derived data products are produced by software. Software comes in versions. As the software improves, the entire Level 0 archive is reprocessed to produce a new edition of the data. Once this edition is published, it becomes part of the science record and must remain published (unchanged) forever. Suppose you have an archive growing at one terabyte per year at level 0 with a derived data product that is 10x smaller. In year 1, the data product is 10% of the storage. In year 5 you have five terabytes of level 0 data and .1+.2+.3+.4+.5 = 1.5 TB of level 1 data. Over the $N-$year life of the project, the level 1 data volume grows as N^2 while the level 0 data grows as N. Even if there is only one data product, the derived data volume will eventually overwhelm the Level 0 data volume. But, there are many Level 1 products - and their number grows with time.

The 5-Year peak: When the archive starts, data arrives at a certain rate, let's say 10TB per year. After 5 years, you will have 50 TB of Level 0 data and perhaps 50 TB more of Level 1 data products. In the mean time, Moore's law will have made storage and processing about ten times less expensive. But in addition, you need to replace equipment that is more than 3 years old. So, the cash flow is a fairly complex combination of these three forces: data inflation, Moore's law, and depreciation. Figure 1 plots these three forces. Ignoring data inflation, the peak is at 2.5 years, but with data inflation it is at 5 years.

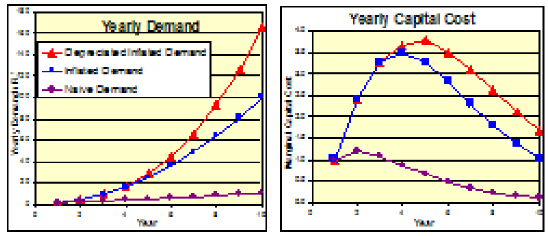


Figure 1: Archive expense (yearly outlay) for an archive where equipment depreciates over 3 years, where Moore's law improves prices by 60% per year, and where data arrives at a constant rate, where the data inflation of ten data products is quadratic.

Overpower by 6x: However much processing and storage you think you need, you need six times as much. The first factor of two is easy to explain: if you store data, you need multiple copies just to be safe. If you do some computation, you should plan to use 50% of the processing power because things inevitably break, and mistakes are inevitably made. So, it is better to build in a factor of 2.

What about that other factor of 3? Well, for storage, you need to have the "production version" and the "next-production version" As you are building the next version you often find yourself sorting the data or indexing it or doing some operation that doubles or triples the size of the data (old version, intermediate version, target version.) You can generally find a way round this, by being careful (and lucky), but it is generally a lot more economical to optimize people's time rather than computer hardware. That is the source of the factor of 3.

For example, when you are reprocessing old stuff *and* current stuff, it is important to have substantial processing, IO, and storage over-capacity so that you can redo the work before it is time to redo the redo.

Automate Data Loading: A substantial part of the project's energy will go into data loading. The database will be reloaded several times - perhaps once or twice a year. So it is essential that the load process be automatic and that there be automatic ways to detect and reconcile data quality errors. People told us this before we started - but we did not believe them. They were right. We invested in this technology late and so wasted a lot of time. At this point, data loading is fairly automatic.

Performance Regression Tests: As the database evolves, and grows, it is essential to have a regression test. The ones we settled on were (1) we should be able to reload the entire database in a week, and (2) none of the 20 queries should take more than 10 minutes to run. Every major release causes us to fix one or another problem and to buy much better hardware in order to meet these performance goals.

Data Pyramid: It is very convenient to have a 1GB, 10 GB, and 100 GB subsets of the database. The smallest database fits on a laptop. It facilitates quick experiments on one or another design issue. The larger databases of course require more equipment to support them, but each has its uses. One particularly useful aspect of the 1GB DB and the corresponding web server is that you can give it to someone as a "double-click-to-install" application. OK, now they have a fully functional personal SkyServer. Now they can throw away the SDSS data and replace it with their data. This has been a very easy way for people to get started with the code.

Capture the Bits: The Sloan Digital Sky Survey is a federation of more than 25 institutions and more than 200 scientists. Much of the system's design has been done with teleconferences and email and source-code-control systems and a bug-tracking database. Today, you can answer almost any question by asking the experts. But the time is coming when the experts will have forgotten or will no longer be with us. So, it is important to curate both the data and all the discussions about how it was gathered and processed. We wish we had the resources to carefully curate all the SDSS design decisions and software. We do not. Short of that, what we are doing is capturing all the bits we can - all the eMail logs, all the software versions, all the bug logs, all the meeting minutes ... everything we can lay our hands on. This is all going into a big repository. Our hope is that future software will be able to organize and mine this information.

7 Summary

We began by commenting that science is changing and that computer science is in a position to help its sister sciences manage the impending data avalanche. One question we sought to answer was: “Can databases help?” The simple answer is: “Yes, if ...” The astronomers for example needed a few things not part of the standard packages (spherical spatial data access, statistical functions,...), and they needed database expertise to help them bridge the gap between their problem and our technology. The sciences also face data mining and data visualization challenges that we have not addressed here (beyond pointing out how important they are.). This is a golden opportunity for computer science to make a contribution if we will take the time to work with our colleagues in X for almost any science X.

Acknowledgments

SkyServer evolved from Tom Barclay’s TerraServer design. SkyServer, SkyQuery, and many other ideas described here were developed by the team consisting of Tamas Budavari, Gyorgy Fekete, Nolan Li, Maria Nieto-Santisteban, William O’Mullane, Peter Kuntz, Don Slutz, Chris Stoughton, and Ani Thakar. A.S. was supported by grants from NSF, NASA, the W.M. Keck Foundation, and the Gordon and Betty Moore Foundation.

References

[CAS] *http://skyserver.sdss.org/cas/en/*.

[Cone] *Cone Search definition*. http://www.us-vo.org/pubs/index.cfm.

[Gray02a] J. Gray, A. S. Szalay, A. R. Thakar, C. Stoughton, J. vandenBerg: *Online Scientific Data Curation, Publication, and Archiving*. SPIE, 22-28 August 2002, Waikoloa, Hawaii.

[Gray02b] J. Gray, A. S. Szalay: *The World Wide Telescope: An Archetype for Online Science*. VLDB Journal, 6(4):333–348, 1997.

[Gray03] J. Gray, A.S. Szalay, A. Thakar, P. Kunszt, C. Stoughton, D. Slutz, J. vandenBerg: *Data Mining the SDSS SkyServer Database*. Distributed Data & Structures, pp 189-210 Carleton Scientific 2003, ISBN 1-894145-13-5.

[Nieto-Santisteban 04a] M. A. Nieto-Santisteban, W. O’Mullane, J. Gray, N. Li, T. Budavari, A. S. Szalay, A. R. Thakar: *Extending the SDSS Batch Query System to the National Virtual Observatory Grid*. MSR-TR-2004-12.

[Nieto-Santisteban 04b] M. A. Nieto-Santisteban, A. S. Szalay, A. R. Thakar, W. J. O’Mullane, J. Gray, J. Annis: *When Database Systems Meet The Grid*. MSR-TR-2004-81.

[SIAP] *Simple Image Access Protocol*. http://www.us-vo.org/pubs/index.cfm.

[Szalay00] A.S. Szalay, J. Gray, P. Kunszt, A. Thakar: *Designing and Mining Multi-Terabyte Astronomy Archives: The Sloan Digital Sky Survey* . Proc. ACM SIGMOD, June 2000, pp. 451-462.

[Szalay02a] A. S. Szalay, J. Gray, J. vandenBerg: *Petabyte Scale Data Mining: Dream or Reality?*. SIPE, 22-28 August 2002, Waikoloa, Hawaii.

[Szalay02b] A. S. Szalay, T. Budavria, T. Malika, J. Gray, A. R. Thakar: *Web Services for the Virtual Observatory*. SPIE, 22-28 August 2002, Waikoloa, Hawaii.

[UCD] *Unified Content Descriptors (UCDs)*. IVOA Proposed Recommendation 26 October 2004, http://www.ivoa.net/Documents/.

[VO Registry] *http://nvo.stsci.edu/voregistry/index.aspx*.

[VO-services] *Web Services for the Virtual Observatory*. http://www.voservices.org/.

[VOtable] *3D VOTable: A Proposed XML Format for Astronomical Tables,*. http://www.us-vo.org/pubs/index.cfm.

A Vision for PetaByte Data Management and Analysis Services for the Arecibo Telescope

Manuel Calimlim[1] Jim Cordes[2] Alan Demers[1] Julia Deneva[2]
Johannes Gehrke[1] Dan Kifer[1] Mirek Riedewald[1] Jayavel Shanmugasundaram[1]
[1]Department of Computer Science and [2]Department of Astronomy
Cornell University
http://www.cs.cornell.edu/database/escience

Abstract

We survey the initial steps of a project to build a data management and data mining system for astronomy data generated by the Arecibo Telescope. The total amount of data that our project will have to manage will approach one Petabyte over five years. We describe some of the scientific challenges from the astronomy side, and we discuss initial thoughts on how to address these challenges through novel data mining research.

1 Introduction

The rapid growth in the generation of digital data is changing computational science in a fundamental way. Traditionally, the scope of computational problems was limited by the available processing power. But today, many problems are extremely data-intensive, and the lack of large-scale storage infrastructure creates a new bottleneck. Thus, modern data-intensive applications need high-performance computational resources *and* a system in which the computational resources are *tightly coupled* with large-scale storage.

At Cornell high-performance computing currently is being integrated with a new large-scale storage infrastructure, which will grow to one Petabyte over the next few years. On top of this system, we are building novel data management and analysis services to support research on finding new pulsars and other interesting celestial objects using data from the Arecibo Telescope.

The Arecibo Observatory (`http://www.naic.edu`), shown in Figure 1 is operated and managed by Cornell University via the National Astronomy and Ionosphere Center under a cooperative agreement with the NSF. The Arecibo telescope is the world's largest radio telescope in terms of collecting area and thus can conduct the most sensitive surveys for time-variable, pointlike objects. A multibeam feed system ALFA (Arecibo L-band Feed Array) has been installed during the summer of 2004. The feed array consists of seven dual-polarization horn feeds that provide 14 simultaneous data streams. The seven feeds point in different, but nearly contiguous directions, allowing a 7-fold speedup of the surveying rate. More details about ALFA can be found at http://alfa.naic.edu.

Pulsars are neutron stars, one of the endpoints of stellar evolution, which are about 10 km in radius and have 1.4 times the mass of the sun. They are, in many ways, giant nuclei. The pulsar surveys will be the deepest

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Figure 1: The Cornell Arecibo Telescope

(reaching to the greatest distances) ever undertaken and are expected to yield not only a large number of new pulsars ($\sim$ 1000) but also exotic objects, including millisecond pulsars spinning near the break-up speed of a neutron star; neutron stars in compact binaries with orbital periods of a few hours or less, and companion stars that are other neutron stars or black holes; and neutron stars moving rapidly owing to their birth in off-center supernova explosions. These discoveries are expected to provide numerous opportunities for followup research on the equation of state of nuclear matter, gravitation physics, and gravitational waves.

The data storage requirements for the P-ALFA surveys are very large ("astronomical"). The raw data from the P-ALFA surveys will amount to about 800 Terabytes, and it will take three to five years to acquire this data.

2 Background and Prior Research

Pulsar surveys not only discover new pulsars, but in particular aim to find exotic objects of interest, including millisecond pulsars with spin periods as fast as 1.6ms and potentially even shorter periods, and compact binaries with orbital periods of a few hours or less. For instance, compact binaries have sufficiently strong gravity that non-Newtonian gravitational theories (such as Einstein's General Theory of Relativity) are needed to account for the orbits, which are precision monitored through continued "timing" observations of a discovered pulsar. Thus, pulsar research provides unique and important opportunities for studying extreme states of nuclear matter, gravitation physics, and gravitational waves. As an example of the physics payoff, a binary pulsar discovered in 1974 at Arecibo by Hulse and Taylor was used to infer the existence of gravitational waves in accordance with Einstein's General Theory of Relativity [HT75]. The Hulse-Taylor binary consists of two neutron stars in an 8-hour orbit with orbital separation smaller than the Sun. For this work, Hulse and Taylor were awarded the Nobel Prize in physics in 1993.

Pulsar search procedures have evolved over the past 30 years according to our growing understanding of astrophysics and populations of compact objects such as neutron stars and black holes, but also thanks to the growth in computational and data storage capacity.

To process the raw data, we apply a filter with three parameters: (a) the temporal period; (b) the dispersion measure, which quantifies how pulses arrive differentially at different frequencies; and (c) the pulse duty cycle, which is the pulse width divided by pulse period. Figure 2 shows a single pulse from a strong pulsar in the Crab Nebula and the differential arrival times owing to dispersion. A trace is shown for each of 256 frequency channels. The slope of the locus describing the arrival time of the pulse in each channel is proportional to the

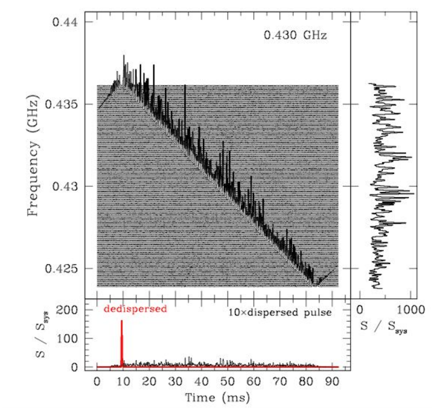


Figure 2: A single pulse from the Crab pulsar in the Crab Nebula

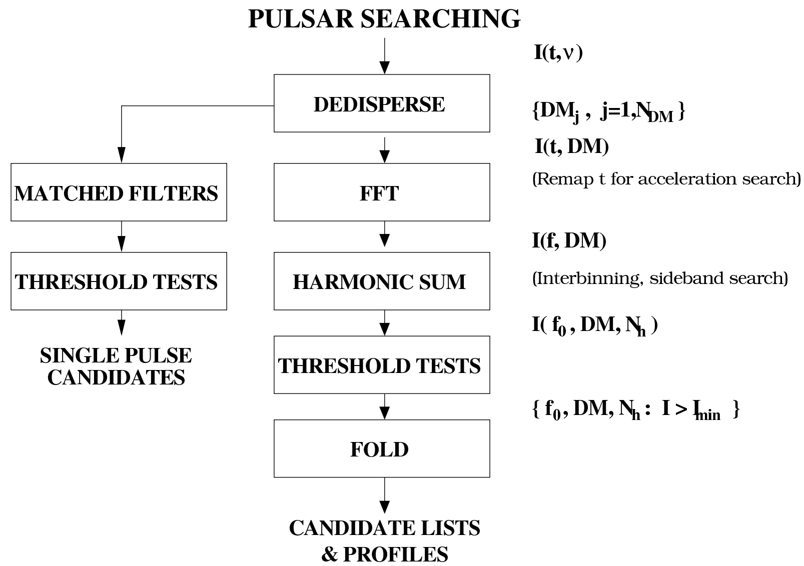


Figure 3: Flow diagram for a pulsar search algorithm

reciprocal of the "dispersion measure" (DM), a quantity that is unique to each line of sight (the endpoints of the pulse show positive slopes owing to aliasing in the sampling system). The bottom panel in Figure 2 shows the sum over frequency channels both with ("dedispersed") and without ("dispersed") compensation for the dispersion effect.

The main problem in pulsar surveys is that we do not know the parameters of the filter in advance. Thus, existing algorithms *search* a large space of possible parameters to find plausible settings that could mark the data as a candidate for a celestial object.

New pulsars are found largely through periodicity searches of pulse trains that are far weaker than the pulse shown. Single pulses are usually buried in noise, so a coherent analysis must be applied to long time series to bring out any periodic signal. While most data sets produce multiple candidate signals, the great majority of these are spurious signals resulting from locally generated radio frequency interference (RFI), one of the main problems of detecting real celestial signals. In order to recognize and remove RFI, we need to perform an analysis of the data products from the entire sky, taking into account that RFI signals repeat (but are episodic), while a single celestial signal should appear from only one direction on the sky. Due to the small size of existing pulsar surveys, current searches for filters could proceed brute-force without any sophisticated search strategies and high-performance algorithm implementations.

As preparation for full ALFA surveys, we have done a pilot survey of one square degree of sky using a single-beam system at Arecibo. The project has made use of pilot-Arecibo data to construct a database and develop analysis tools for investigating candidate signals from pulsars that will warrant reobservation. Reobservations are needed to robustly confirm the reality of pulsars versus artificial radio frequency interference, which is a growing problem that requires new filtering algorithms.

3 The P-ALFA Surveys

The proposed surveys for pulsars include (a) mapping of the entire Galactic plane of the Milky Way visible with Arecibo (within ± 5 deg of the midplane) where the greatest concentration of pulsars is expected; this represents a swath of sky equal to about 500 square deg, and (b) searching further out of the Galactic plane (e.g. ± 15 deg) in a shallower survey to find millisecond pulsars and binary pulsars that are expected to have a thicker population distribution than the bulk of pulsars.

Why search this much sky? The survey payoff comes in two forms. First, the number of detected rare, exotic objects scales with the expected total number (N) of new pulsar discoveries. Based on previous surveys, having N = 1000 implies that we will find a handful of exotic binaries, etc. that will provide the greatest payoff in basic physics. Second, the total sample can be used to map the ionized gas and magnetic field of the Galaxy. Pulsar signals carry information about intervening magnetized, ionized gas because the signals are distorted by the consequent variable index of refraction. Techniques are well known for measuring these distortions and thus quantifying the integrated (along the path) electron density and product of electron density and line-of-sight component of magnetic field. The basic effects are known as dispersive arrival times (similar to delays in an optical fiber) and Faraday rotation of the plane of polarization of the electromagnetic waves. The challenge in using such integrated measures is to invert them into information about the gas density and magnetic field at each location in the Galaxy. Such inversion is similar to tomographic imaging. Generating tomography-based maps of gas properties is much more challenging than making the basic measurements for each line of sight. Examples to date may be found in [TC93], [CL02] and [WCK$^+$03].

The raw data will be subjected to a novel multistep data analysis for identifying real celestial signals with expected (and hopefully new) signatures. The multiplicity of signals from the P-ALFA surveys drives data rates upwards and provides the opportunity for applying more intricate filtering algorithms that distinguish celestial signals from locally generated RFI. To realize the great potential of the ALFA system, new data mining algorithms are needed for tackling RFI and for detecting celestial objects of greatest interest.

The 14 signals from the "front-end" feed system (ALFA) will be signal processed in real time using a fast digital spectrometer that provides the basic "dynamic spectrum" $I(t, \nu)$, intensity vs. time and radio frequency, for each of the seven feed antennas. (The two polarization channels will be summed in real time, yielding seven distinct dynamic spectra.) Prior to the search analysis, the raw data will be filtered to remove radio frequency interference, which is expected to be nearly identical in all seven beams, whereas celestial signals will appear in at most 3 beams. As shown in Figure 3, each of the seven dynamic spectra is subjected to a detailed Fourier and statistical analysis whose results are winnowed to provide candidate periodic pulsar signals and candidate single pulse events. The dynamic spectra for each telescope pointing comprise a size equal to 7 beams $\times$ 1024 frequency channels $\times$ 10^7 time samples (approximately, since survey dwell times are now being optimized). The time duration for each pointing will be $\sim$ 300s. Each dynamic spectrum will be searched for about 10^3 separate values of dispersion measure, yielding $7{\times}10^3 \times 10^7 \times 32$ separate statistical tests, where the 32 factor is the number of separate harmonic sums that are calculated, or $\sim 2 \times 10^{12}$ separate tests. Thresholds are set so that, per beam, we will have only 10 to 10^2 events above threshold that require further investigation. *Subsequent to* the search analysis performed on each beam, candidate signals found in the 7 beams will be cross compared to remove low-level RFI that was missed previously, thus improving the quality of candidate lists. Additional complexity of the analysis involves optimization against potential motion of a pulsar in an orbit. To do so requires remapping the time series according to trial *accelerations* using typically about 200 trial values. Single-pulse candidate searches represent about 10% of the computations of the periodicity searches but add significantly to data products that need to be investigated in the meta analysis of all telescope pointings. Altogether, about 25,000 telescope pointings will be made to cover the Galactic plane with a similar number to cover portions of sky outside the plane. With the 300 s dwell time per pointing, it will take $\sim$ 2100 hr and $\sim$ 800 TB to acquire and store the raw data for the Galactic plane search.

4 Data Mining Challenges

Astronomy data provides an exciting opportunity to test and develop new data mining techniques because of the wealth of information hidden in the raw data. The raw data is produced in such large quantities that *efficient* data mining and analysis becomes very challenging.

Common data reduction techniques such as subsampling, smoothing, and functional approximation are either not appropriate or need significant improvements to produce results of the same caliber as in other domains. The reason for this is the low signal-to-noise ratio. It is true that some of the signals are stronger than the noise. These include radio frequency interference from local sources as well as strong pulses from some pulsars. However, many interesting signals are *weaker* than the noise. This is the case for the majority of pulsars. The pulses they produce are weak and appear as spikes in signal intensity. Subsampling can easily miss the peaks of these spikes. Furthermore, subsampling is not advisable in noisy domains for statistical reasons — a smaller sample yields less accuracy. Careless use of smoothing and functional approximation can also cause problems.

Detection of astrophysical signals in data from Arecibo involves matched filtering, which optimizes the signal-to-noise ratio of a test statistic. Astrophysical signals are always corrupted by additive noise and generally are themselves stochastic in nature. Matched filtering requires knowledge of the functional form of the signal. For some aspects of the signals, such as the dispersion effect and periodicity mentioned before, we know these forms. However, other aspects of pulsar signals are not so well known *a priori*, such as the detailed pulse shape. In addition, we wish to discover entirely new classes of objects in the Arecibo data. These may include variations on the types of neutron stars, such as very sporadic emission, whose detection cannot rely on the periodicity, and they may include pulse shapes dissimilar to those of known pulsars. In addition, we would like to discover radio emission from other kinds of objects, such as black holes and extrasolar planets. In these cases, we know very little about the signature of the signal. For this reason, complete mining of the data requires that we search for signals with unknown structure.

A typical approach is to start with a collection F of smooth functions such as splines and find the $f \in F$ that minimizes:

$$\left(\sum_i (d(i) - f(i))^2\right) + \lambda \int \left(f''(x)\right)^2 \, dx$$

where d is the time-series and λ is a smoothness parameter [RS02]. Note that the second term is a roughness penalty that reduces the amount of wiggle in the approximating function by favoring functions with a smaller second derivative (acceleration). This suggests that information will be lost when we approximate a sharp spike (that may lack a second derivative or have an extremely large second derivative at some points) with a smooth function. Furthermore, because the signal is weaker than the noise, we may end up averaging the signal with the noise. This will spread out the spike (in time) and weaken its intensity.

The same can happen if we try to approximate the signal with a Fourier series. To reduce data size, the Fourier approximation (FFT) requires removing terms whose coefficients in the Fourier expansion are small. Thus the FFT is currently not used as a means of data reduction, instead it is used as a tool to discover the rotational frequency of a pulsar. Indeed, when searching for pulsars, *all* Fourier coefficients are kept and a technique called *harmonic folding* is used to discover the rotational frequency (in harmonic folding, the coefficients of several harmonics of a base frequency are added together). This yields better results than simply looking at the frequencies with highest Fourier coefficients.

To extract weak signals from the noise, it is essential for data mining to incorporate background knowledge in a principled way. Information such as "the signal is periodic" or "the signal, if present, has the following characteristics or the following functional form" must be handled automatically and efficiently. How can we use background knowledge to boost efficiency and/or accuracy of data mining techniques? Right now this is done an a case-by-case basis: Fourier series and harmonic folding for periodic signals, time warping the time-series to "normalize" it if the signal has a parametric form. Is there a more general language (short of a programming language) that can be used to specify certain background knowledge and from which algorithms can be automatically constructed? This is an open question in data mining today and astronomy data is a promising test-bed and motivator for ideas in this direction.

Closely related to the concept of mining with background knowledge is the notion of related mining tasks. Consider the following three tasks: detecting pulsars that periodically emit weak pulses, detecting pulsars that

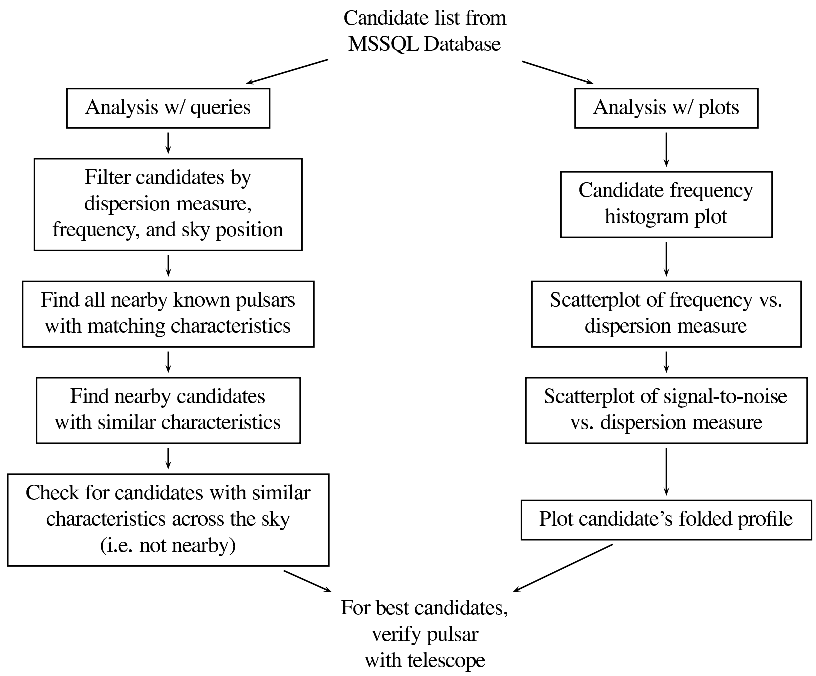


Figure 4: Pulsar candidate online analysis

occasionally emit strong signals, and detecting solar bursts. Intuitively the two pulsar-detection tasks are related to each other but not to the detection of solar bursts. The processing required for a task can be described as a sequence of steps (removal of RFI, dedispersion, FFT, harmonic folding, etc.) and each step generates intermediate data such as dedispersed time-series, lists of RFI signals, lists of Pulsar signals, and so on. Related tasks can share information and intermediate data representations to improve throughput in a way that is analogous to the sharing of buffer pages between related queries in a database management system (DBMS). For example, almost all tasks can share dedispersed data and RFI signals (which would then be removed from the time series). Other types of signals can also be shared between tasks so that they can be removed from the time series. The relationship to a DBMS is not superficial. There is a potential and need for a Data Mining Management System (DMMS) to identify related tasks, to identify shareable information and intermediate data representations, to manage this information (on disk) for optimizing future queries, and to incorporate background knowledge into the mining process.

After processing is finished, results are presented to the astronomer for further analysis. In the case of pulsars, the result would include a graph of the average pulse profile. At this point the astronomer needs to decide what is interesting, what isn't, and what deserves more investigation. The amount of results can be very large and currently results are thresholded based on signal strength. Here we have opportunities for more applications of classical machine learning and data mining. The final results can be clustered, classified, and ranked based on preferences derived from human interaction. Not only will this reduce the burden on the astronomer, but it will provide more testing data and potential benchmarks for extensions to classical mining.

5 The Current Data Management and Analysis System

The initial pilot survey that focused on a small portion of the sky is online and ready for analysis. The output of analyzing the raw signal data is divided into two categories of processed data: pulsar candidates and single giant pulses. Pulsar candidates are usually identified by the frequency of periodic radio pulses while single giant

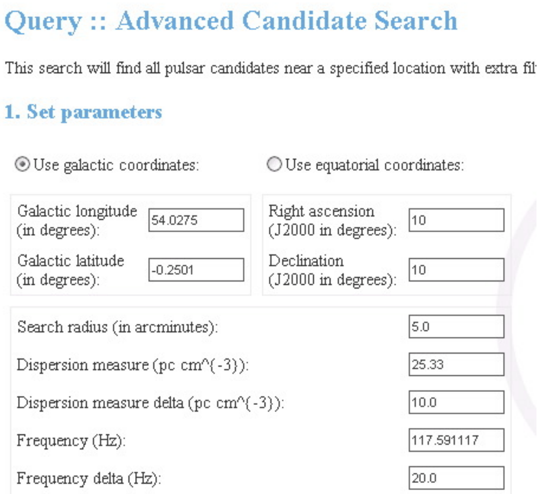


Figure 5: Advanced Candidate Query example

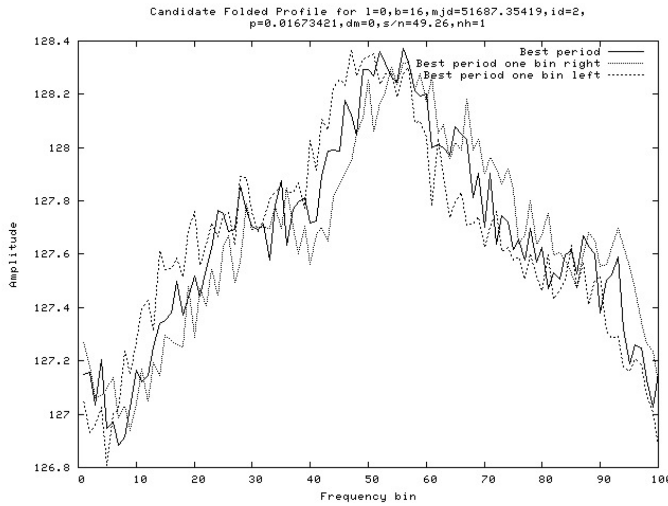


Figure 6: Candidate pulse profile example

pulses are any pulses above a certain threshold.

Most of the analysis for pulsar candidates is geared towards identifying and removing radio frequency interference (RFI). Figure 4 shows a typical workflow for online analysis of candidates. At the top of the figure, all of the candidate information is loaded into a relational database for fast searching of the data. The left branch shows some sample queries that can be made from the website to reduce the number of candidates.

First, a simple filter query can easily identify and reject some common sources of RFI. Dispersion measure is used to calculate the distance of the radio object. When the dispersion measure is close to zero, the signal is most likely terrestrial in origin. Also, the candidate list can be pruned with restrictions on frequency; removing candidates that match the frequencies of well-known sources of RFI eliminates a large number of invalid pulsar candidates.

After the obvious RFI is detected, more advanced queries are employed. Pulsar candidates are compared to the list of known pulsars to avoid detecting an object that has already been discovered. Then there is a search for matching candidates in nearby pointings since good pulsar candidates with strong signals will frequently appear in adjacent pointings. An advanced candidate search with deltas over distance, frequency, and dispersion measure is displayed in Figure 5. Finally, a search for matching candidates in far pointings is conducted; a pulsar candidate that appears all over the sky is likely RFI since a pulsar's signal should come from one direction instead of all directions.

A number of online plots are also available for the user. The user can plot a histogram of the candidate frequencies of a particular pointing versus the frequencies across all pointings. This can help to determine if an event seen in a pointing is a common RFI type event. Two scatterplots (one with frequency vs. dispersion measure and the other with signal-to-noise ratio vs. dispersion measure) are used to quickly check for trends in the data that would indicate RFI contamination. The most important plot is the folded pulse profile for pulsar candidates (see Figure 6). The folded profile helps classify the object and usually is the final determination for which objects are actual pulsars.

The best pulsar candidates are saved in a catalog. The raw data for these pointings is re-processed and re-analyzed to verify the possible pulsar. The very best candidates need further observations with the telescope to confirm the reality of new pulsars versus possible artificial radio frequency interference.

Analysis of single giant pulses is similar to the pulsar candidate analysis. Figure 7 shows the online tools available. Note that only a basic filter query is supported. Because single giant pulses are comprised of any radio pulse above a threshold, the number of giant pulses is orders of magnitude larger than the number of pulsar

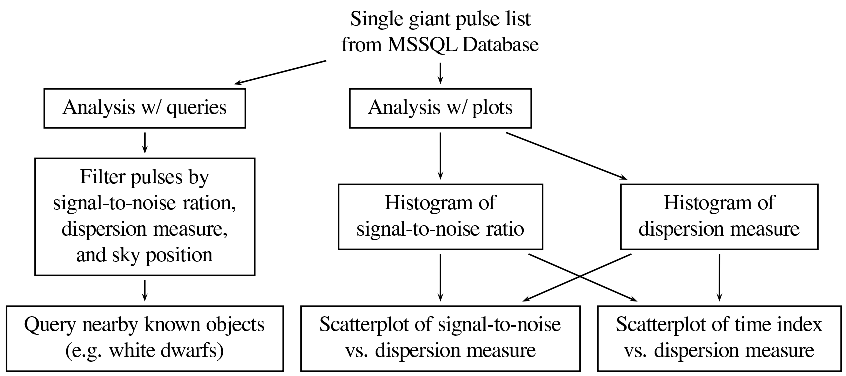


Figure 7: Single giant pulse online analysis

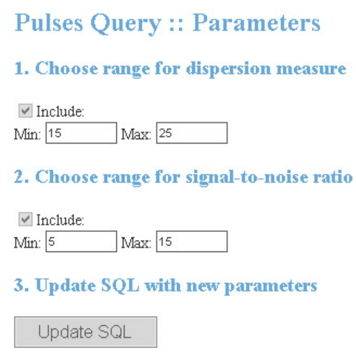


Figure 8: Candidate pulse profile example

candidates. Generally, this means that complex queries run very slowly over the single giant pulses. Special spatial indices have been added to the database to speed queries over location, but performance for the single pulses is a major issue.

Filtering the giant pulses by signal-to-noise ratio, dispersion measure, and sky position helps identify interesting single pulses, i.e., pulses with a high signal-to-noise and dispersion measure. Figure 8 shows the webpage interface that specifies the parameter ranges for searching the single pulses. Once interesting pulses are found, users can query for nearby objects of interest. These include white dwarfs, gamma ray sources, x-ray sources, and other radio sources that may explain the single giant radio pulse.

The analysis plots for the single pulses also help isolate important phenomena. Histograms for signal-to-noise and dispersion measure should reveal spikes of interest. Scatterplots with signal-to-noise vs. dispersion measure or time index vs. dispersion measure are used for recognizing which pointings have too much RFI.

The single pulse analysis can reveal pulsars that the candidate analysis misses. In fact, since so much data is captured, more exotic objects than pulsars may be found (e.g., compact binaries or neutron stars). The potential for data mining over the single giant pulses is broader and may be the most interesting area for future research.

6 Expected Results: One Example

Recall the discovery of the Hulse-Taylor binary, which consists of two neutron stars in an 8-hour orbit. In the last 30 years, only five additional binaries have been found because they are scarce and require a great deal of telescope and computer time. The orbits of double neutron star binaries (and any neutron-star/black-hole binaries) are shrinking because of energy losses associated with the gravitational waves. Consequently, we expect the Galaxy to contain binaries with orbital periods *much smaller than 8 hours*. Short-period binaries are difficult to detect because traditional search algorithms rely on the Fourier analysis of equally-spaced (or nearly so) pulses (see [Cor02] for an overview); Doppler shifts from orbital motion reduce the sensitivity of such searches unless compensating, computationally intensive algorithms are used [RCE03]. Processor speeds now are just beginning to make comprehensive searches for binaries feasible. We expect that important new binary systems will be found in ALFA surveys that will be far richer than the Hulse-Taylor binary in providing opportunities for testing gravitational physics.

7 Future Directions

As mentioned before, our goal is to mine the Arecibo data for both known classes of signals — periodic, dispersed pulsar emissions and single dispersed pulses from pulsars — and new classes of objects. The starting point is the dynamic spectrum and each kind of signal will produce a certain characteristic pattern in the frequency-time plane of the dynamic spectrum. Our aim is to develop methods that will allow us to detect (i.e. distinguish from noise) and classify all non-noise events according to their frequency-time morphology, followed by classification in terms of what kind of terrestrial interference or kind of celestial source is involved. This overall aim requires that we fully exploit the growing database by developing tools that are both open-ended and efficient.

8 Acknowledgments

Jim Gray gave us helpful advice throughout many discussions. Alex Szalay offered us generous support through usage of his software and through a visit to his group. David Lifka and his team at the Cornell Theory Center provide invaluable support to configure and run the system.

Our system is supported by a Research Infrastructure Grant Number CNS-0403340 from the National Science Foundation, and by an E-Science grant from Microsoft Corporation. Johannes Gehrke's data mining research has been supported by the National Science Foundation under Grants IIS-0084762 and IIS-0121175. Jim Cordes's research has been supported by NSF Grants AST-0206035 and AST-0206036. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

References

[CL02] J. M. Cordes and T. J. W. Lazio. NE2001.I. A New Model for the Galactic Distribution of Free Electrons and its Fluctuations. *ArXiv Astrophysics e-prints*, July 2002.

[Cor02] J. M. Cordes. Pulsar Observations I. – Propagation Effects, Searching Distance Estimates, Scintillations and VLBI. In *ASP Conf. Ser. 278: Single-Dish Radio Astronomy: Techniques and Applications*, pages 227–250, December 2002.

[HT75] R. A. Hulse and J. H. Taylor. Discovery of a pulsar in a binary system. *Astrophysical Journal Letters*, 195:L51–L53, January 1975.

[RCE03] S. M. Ransom, J. M. Cordes, and S. S. Eikenberry. A New Search Technique for Short Orbital Period Binary Pulsars. *Astrophysical Journal*, 589:911–920, June 2003.

[RS02] J. O. Ramsay and B. W. Silverman. *Applied Functional Data Analysis*. Springer-Verlag, New York, 2002.

[TC93] J. H. Taylor and J. M. Cordes. Pulsar distances and the galactic distribution of free electrons. *Astrophysical Journal*, 411:674–684, July 1993.

[WCK+03] J. M. Weisberg, J. M. Cordes, B. Kuan, K. E. Devine, J. T. Green, and D. C. Backer. Arecibo 430 MHz Pulsar Polarimetry: Faraday Rotation Measures and Morphological Classifications. *ArXiv Astrophysics e-prints*, October 2003.

Workload-Driven Schema Design for Large Scientific Databases

Stratos Papadomanolakis, Anastassia Ailamaki
Carnegie Mellon University
{stratos,natassa}@cs.cmu.edu

Abstract

Database applications that use multi-terabyte datasets are becoming increasingly important for scientific fields such as astronomy and biology. Scientific databases are particularly suited for the application of automated physical design techniques, because of their data volume and the complexity of their workloads. Current automated physical design tools focus on the selection of indexes and materialized views. In large-scale scientific databases, however, the data volume and the continuous insertion of new data allows for only limited use of such redundant structures. In this paper we propose schema design techniques to improve query performance without redundancy. We present AutoPart, an algorithm that automatically partitions database tables utilizing knowledge of a representative workload. To evaluate AutoPart we built an automated schema design tool that interfaces to commercial database systems. We experiment with a real world astronomical database, the Sloan Digital Sky Survey. We demonstrate the benefits of partitioning for large-scale systems: Partitioning alone improves query performance by a factor of two on average. With indexes, the partitioned schema outperforms the original schema by 20% for queries and a factor of five for updates, while using only half the original index space.

1 Introduction

Scientific experiments in fields such as astronomy and biology typically require accumulating, storing, and processing very large amounts of information. The ongoing effort to support the Sloan Digital Sky Survey (SDSS) [9, 14] provides a comprehensive example for both the terabyte-scale storage requirements and the complex workloads that will execute on future database systems. Similarly, the Large-aperture Synoptic Survey Telescope (www.lsst.org) dataset is expected to be in the scale of petabytes (the data accumulation rate is calculated at 8 terabytes per night). Typical processing requirements on these datasets include decision-support queries, spatial or temporal joins, and versioning. The combination of massive datasets and demanding workloads stress every aspect of traditional query processing.

In environments of such scale, query execution performance heavily depends on the underlying physical design. Recent database research has produced tools that utilize workload information to automatically design indexes or materialized views [1, 4, 6] and automated design algorithms are already incorporated in commercial systems. In the case of large-scale applications like SDSS, performance is optimized through implementing a large set of *covering* indexes, since accessing the large base tables (even through non-clustered indexes) is prohibitively expensive. Large numbers of indexes, however, are expensive to store, as they *replicate* data columns

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

from the base tables. This 'data replication' incurs a significant storage management overhead, especially for multi-terabyte scientific databases. Indexing also complicates updates. For instance, new experimental or observation data is often inserted in the database and derived data is recalculated using new models. During update operations, new (or updated) data values must be sorted and inserted in every index that contains them, with a significant processing and I/O cost. The storage and maintenance overhead caused by redundant, auxiliary structures like indexes or materialized views motivates the development of physical design techniques that improve query execution performance while reducing the overheads.

We explore schema design as way to improve performance without redundancy. We present AutoPart [13], an algorithm that partitions the tables in the original database according to a representative workload. AutoPart is incorporated into an automated schema design tool that interfaces to commercial database systems. By first designing a partitioned schema and then building indexes on the new database, queries can scan the base tables efficiently as well as a smaller set of indexes, thereby alleviating unnecessary storage and update overhead. Because data partitioning increases spatial locality, it improves memory and disk system performance when the covering index set cannot be built due to storage or update constraints.

Our experimental results confirm the benefits of partitioning: Even without the use of indexes, a partitioned schema can speed up query execution by almost a factor of two when compared to the original schema. Combined with indexes, the new schema also outperforms the indexed original schema by 20% (for queries) and a factor of five (for updates), while using only half the original index space.

This paper is structured as follows: Section 2 summarizes related work. Section 3 discusses the partitioning problem in greater detail, while in Section 4 we present the AutoPart algorithm. Section 5 discusses the AutoPart architecture. Section 6 presents our experimental results and we summarize in section 7.

2 Related Work

In this section we present previous work on i) partitioning techniques and ii) automated physical design.

2.1 Partitioning Algorithms

Vertical partitioning is known to optimize I/O performance since the early days of relational databases [15]. To determine a suitable partitioning for a relation, initial studies exploited attribute affinity, a measure of how often queries reference combinations of attributes [8, 10, 11]. Affinity-based approaches produce reasonable results, but they dont use estimates of query costs. Subsequent work introduced such cost models in the partitioning process: An integer programming formulation of the partitioning problem [5] makes use of simplified analytical cost formulas, while a partitioning implementation using pair-wise attribute merging [7] uses a cost estimation module similar to todays query optimizers. For horizontal partitioning, a set of disjoint horizontal fragments can be specified by using the predicates in the workload [2]. The fragments are then combined to minimize an application-dependent cost function. An idea similar to the categorical partitioning technique presented in this paper, is used for the more efficient execution of group queries [3].

2.2 Automated Tools for Index and Materialized View Selection

The method of choice for modern, state-of-the-art automatic design tools is the combination of heuristic search methods with the system's own query optimizer. Index selection tools for relational databases [1, 4, 6] and the automatic declustering techniques for parallel DBMS [12] are based on the optimizers cost estimates. The index selection problem is closely related to vertical partitioning: since the base table structure is perceived as a static property of the database, a viable alternative for reducing the I/O requirements of a query workload is the use of covering, multi-attribute indices to facilitate index-only data access. Such indexes essentially are ordered vertical partitions. The only difference is that indexes are redundant structures, therefore a popular column is replicated multiple times in the final design. Given that the original relations are not necessarily optimized for a particular workload, these tools are often face a difficult problem, since the use of every additional index increases update overhead and data redundancy.

3 Workload-Driven Data Partitioning

We start with a formulation for the vertical partitioning problem: Given a set of relations R = $\{R_1, R_2,...,R_n\}$ and a set of queries Q = $\{Q_1, Q_2,..., Q_m\}$ generate a set of fragments F = $\{F_1, F_2,...,F_N\}$ such that:

- Every fragment F_i stores a subset of the attributes of a relation in R plus an identifier column.
- Each attribute of a relation in R, is contained in exactly one fragment in F. (except for the primary keys).
- The sum of the query costs when executed on top of the partitioned schema, *cost*(Q, (R-F)) is minimized.

We expect the minimal workload cost to be much less than the original cost, because in the partitioned schema queries will avoid accessing attributes they do not need. The Query Access Set (QAS) of a query Q_i with respect to a relation R_j (QAS(Q_i,R_j)), is the subset of R_j's attributes referenced by Q_i. In the ideal case, the QAS for all the queries in the workload are disjoint: then the solution to the vertical partitioning problem is to simply generate a fragment F for each distinct QAS in the workload. Then, each query would have to access a single fragment containing exactly the attributes it references, resulting in minimal I/O requirements. Realistically, however, workloads contain overlapping QAS, making such 'clean' solutions infeasible.

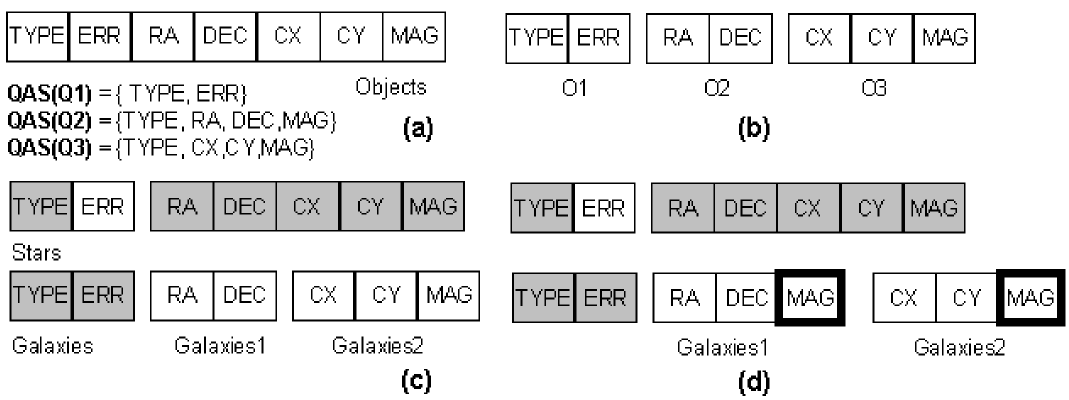


Figure 1: Partitioning example. (a) Original schema. (b) After vertical partitioning. (c) After vertical & categorical partitioning. (d) Column replication.

Consider the example in Figure 1, drawn from a simplified astronomical database. Our database consists of a single table (Objects) that stores objects like galaxies and stars. Our workload consists of queries Q_1,Q_2,Q_3. Figure 1(b) shows one possible solution for vertically partitioning Objects into fragments O_1, O_2, O_3. Q_1 needs to access only O_1, minimizing its I/O requirements. Since the attribute TYPE exists in all QAS however, queries Q_2 and Q_3 will have to access fragment O_1 in addition to O_2 and O_3 and perform additional joins. Also, Q_2 will have to access fragment O_3 to obtain its missing attribute. The alternative approach of merging some of the O_1,O_2,O_3 would result in lower joining overheads, but the queries would have to access a larger number of additional attributes and the I/O cost would increase.

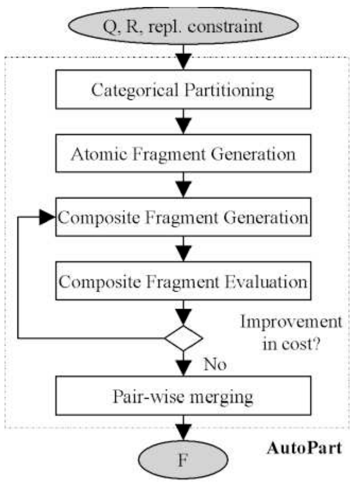
Overlapping QAS in a workload make it impossible to avoid additional joins. Often, however, much of the overlap implied by overlapping QAS is not real. In the previous example, assume that Q_1 restricts its search to objects of type 'Star' (TYPE = 'Star'), whereas Q_2 and Q_3 only care about objects of type 'Galaxy' (TYPE =

’Galaxy’). In this case, considering only QAS leads to ’false sharing’ as Q_1 will process a completely disjoint set of tuples than Q_2 and Q_3. By *categorically* partitioning Objects we remove the overlap between QAS(Q_1) and {QAS(Q_2),QAS(Q_3)}, since the queries now access only the categorical fragments (Figure 1 (c)). The fact that Q_1 needs to access attributes {TYPE, ERR} together does not affect Q_2, Q_3. With this form of partitioning queries benefit from the elimination of unnecessary accesses to objects of the wrong class and also from the removal of categorical columns (in this example we can remove TYPE from the categorical fragments altogether).Note that even in the categorically partitioned schema of Figure 1 (c), there is still an overlap between QAS(Q_2) and QAS(Q_3) on MAG. We reduce the impact of such overlaps, that cannot be removed by categorical partitioning, by allowing the replication of attributes belonging to the intersection of two or more QAS. In our example, we replicated attribute MAG in the two fragments, Galaxy1 and Galaxy2, in order to remove the remaining joins (Figure 1 (d)). The resulting schema has no additional joins or unnecessary data accesses.

4 The AutoPart Algorithm

We define a *fragment* of a relation R to be a new relation containing a subset of R’s attributes. We distinguish between two kinds of fragments: *atomic* fragments are the ’thinnest’ possible, and are accessed atomically in the workload: there are no queries that access only part of an atomic fragment. In addition, atomic fragments are disjoint and their union is equal to R. We define a *composite* fragment as the union of two or more atomic fragments. AutoPart generates a set of atomic or composite fragments than minimizes the input workload’s cost.

The general structure of AutoPart is shown in Figure 2 (a). The first step is to categorically partition the input relations. Section 4.1 describes our categorical partitioning algorithm in more detail. Then the algorithm vertically partitions the categorical fragments. Section 4.2 describes our vertical partitioning approach.




```
categorical_partitioning(relation R, queries Q, size N)
    1. A := categorical attributes ∈ R.
    2. Let X = collection of predicates in Q of the form x_i:{A_i ∈ d ⊂ D_i}
    3. XM := complete_minimal_set({x_i})
    4. horizontal fragments Y := minterm_fragments(X)
    5. if |Y| ≤N then
       5a. let A := A - {a_i} /* remove attribute ai */
       5b. Yi := merge(Y,a_i) /* merge the hor. fragments defined using ai*/
       5c. Select ai that leads in the minimal |Yi|
       5d. Set Y := Yi
6. Repeat step 5 until |Y| satisfies size constraint
7. For each fragment F ∈ Y, remove all attributes in F that take a single value.
```

(a) (b)

Figure 2: (a) Outline of the AutoPart algorithm, (b) Categorical Partitioning algorithm

4.1 Categorical Partitioning

The categorical partitioning step (Figure 2 (b)) generates horizontal fragments of the partitioned relations. The partitioning depends on the existence of categorical attributes in the relations and in the workload. Categorical attributes are attributes that take a small number of discrete values and are used to identify classes of objects. The algorithm first identifies the set of categorical attributes A_i in the relation R and their corresponding domains D_i (step 1), by consulting the system’s designer or the system catalog. Each query in workload Q containing predicates on those attributes, defines a horizontal subset of R, containing all the objects that satisfy the predicates. The purpose of the algorithm is to determine a suitable collection of non-overlapping subsets, which will be assigned to different horizontal fragments. We use the methodology developed in [2].

We can express every query predicate involving each of those attributes in the form $x_i : \{A_i \in d \subset D_i\}$. Let X = $\{x_i\}$ be the collection of such predicates, and assume that it is minimal and complete, according to [2]. Then, the min-term predicates Y(X) [2] computed in Step 4 define a collection of non-overlapping horizontal fragments that can be used to define the horizontal fragments of R. If there exist categorical attributes A_i that take a unique value in the horizontal fragments determined, they are removed (Step 5).

4.2 Vertical Partitioning

The vertical partitioning algorithm first examines an initial solution consisting only of atomic fragments. Using only atomic fragments probably has a negative impact on performance, because of the large number of additional joins that must be performed.

The algorithm improves the initial schema by forming composite fragments that reduce the joining overhead but increase I/O cost: queries accessing a composite fragment don't necessarily reference all the attributes in it. The Composite Fragment Generation module determines a set of composite fragments that should be considered for inclusion in the schema, while the Composite Fragment Selection module evaluates the available options and chooses the fragments that are found to provide the highest improvements for the workload.

AutoPart iterates through the composite fragment generation and selection steps, each time expanding the fragments selected in the previous steps. The composite fragments considered may contain attributes that are also included in other fragments in the partitioned schema, thus allowing for attribute replication. The storage space available for replication can be controlled by an *storage constraint* parameter provided to the algorithm. The storage constraint that maximizes performance depends on the properties of the workload. One can identify the optimal value for the storage constraint by running AutoPart multiple times.

When the workload cost cannot be further improved by the incorporation of composite fragments, the resulting schema is passed through a sequence of pair-wise merges of fragments, attempting to further improve performance. A detailed description of AutoPart's components can be found in [13].

5 System Architecture

This section describes the functional blocks of the automated schema partitioning tool, depicted in Figure 3. The system was implemented using Java (JDK 1.4) and JDBC and the DBMS is SQL Server 2000.

- QUERY PARSER. This module receives as input the original queries (Q) and the tables to partition (R). Its output is the queries in a parsed representation (Q_P).
- TABLE DESIGNER. The Table Designer module is the heart of the schema design tool. It receives as input the set of parsed queries (Q_P) and the original schema definition (W_{ORIG}), and applies the vertical partitioning algorithms of Section 4. Its output is a set of candidate partitioned schemas (W_{PART}) to be evaluated by the query optimizer.
- QUERY REWRITER. The rewriter uses each partitioned schema definition (W_{PART}) and the set of parsed queries (Q_P) to produce a set of equivalent rewritten queries (Q_R) that access the fragments in W_{PART}.
- DBMS INTERFACE. This is a JDBC interface to the database currently hosted by the SQL Server. The interface executes table and statistics creation statements according to W_{PART}. To accurately estimate query costs, our tool provides the query optimizer with the correct table sizes and statistics for the partitioned schema. We found that in order for the virtual and real cost estimates to agree, the statistics must be generated using full data scans and not by random sampling.

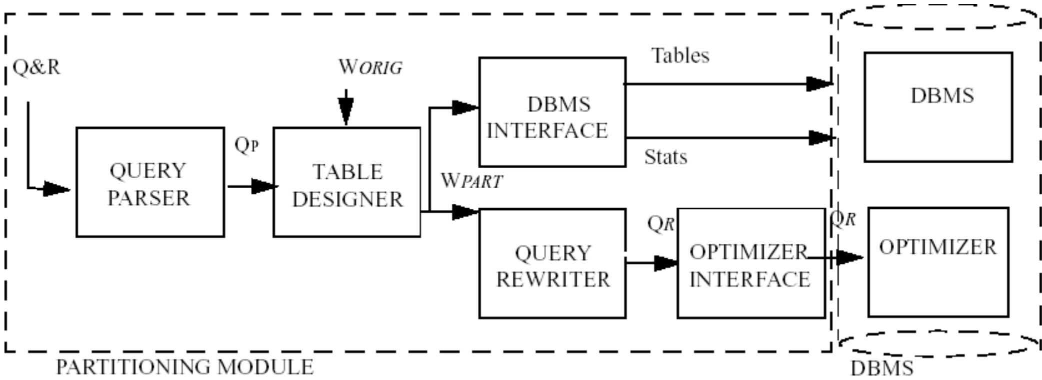


Figure 3: Partitioning tool architecture

- SYSTEM CATALOG. The DBMS catalog stores information like table sizes, row sizes and statistics. To facilitate query cost estimation, we update the system catalog tables with information reflecting the new schemas, without actually loading the partitioned tables with data.
- OPTIMIZER INTERFACE. This JDBC interface receives as input the rewritten queries (Q_R) and uses the query optimizer to obtain query plan information and cost estimates.

6 Experimental Evaluation

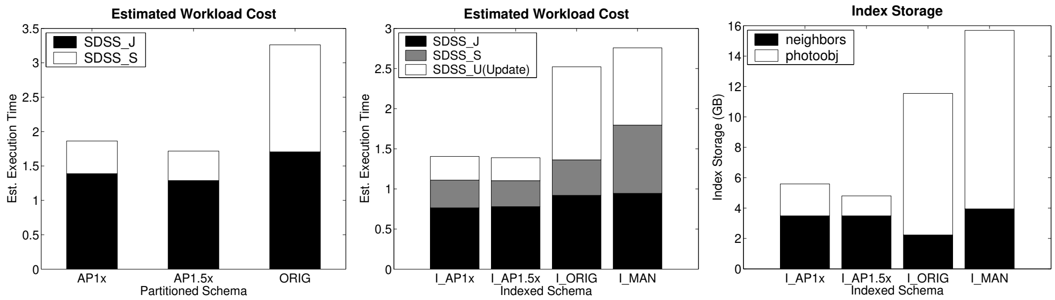
We present experimental results on (a) the performance of AutoPart without indexes and (b) the benefits of partitioning in the presence of indexes and updates.

6.1 Experimental Setup

Our experiments use the Sloan Digital Sky Survey (SDSS) database [9, 14], running on SQL Server 2000. The machine we use for the experiments is a workstation with a single 1-GHz Athlon CPU, 512 MB of main memory and two 120-GB IDE disks. The two largest tables in the SDSS database are *PHOTOOBJ* (22GB), a 'catalog' table with hundreds of attributes for each astronomical object, and *NEIGHBORS* (5GB), which stores spatial relationships between neighboring objects. The SDSS workload [9, 14] is a set of 36 representative SQL statements. In order to simulate the addition of new data, we added two insertion (*SQL INSERT*) statements. *PHOTOOBJ* and *NEIGHBORS* are almost exclusively responsible for the I/O costs in the SDSS workload and we therefore use AutoPart only on them. In the following sections we present performance results in terms of the optimizer estimates of the workload's cost. Table 1 documents the alternative schemas we experimented with. We detail our experimental setup in our recent study [13].

ORIG	Original non-partitioned schema without indexes.
AP1.5x	AutoPart schema with column replication (allowing 50% additional space).
AP1x	AutoPart generated schema without column replication.
I_ORIG	Indexed non-partitioned schema (using the Index Tuning Wizard)
I_MAN	Manually indexed non-partitioned schema
I_AP1.5x	Indexed AutoPart schema with column replication (using the Index Tuning Wizard)
I_AP1x	Indexed AutoPart schema with no column replication (using the Index Tuning Wizard)

Table 1: Description of the schemas used in our experiments



(a) (b) (c)

Figure 4: (a) Workload costs without indexes, (b) With indexes, (c) Storage allocated for indexes

6.2 Evaluation of Partitioning

This section demonstrates that AutoPart significantly improves query execution without the use of any indexes. We derive two partitioned schemas, AP1.5x and AP1x, with and without replication respectively. In the attribute replication case, we set a storage bound for the replication columns equal to 1/2 the original database size. Experimenting with higher storage bounds did not further improve performance.

We distinguish between two groups of SDSS queries. The join-bound group (SDSS_J) consists of four queries, whose execution is bounded by expensive joins among several instances of *PHOTOOBJ* and *NEIGHBORS*. The scan-bound group (SDSS_S) includes queries dominated by table scans. This distinction is necessary because the two query classes do not benefit equally from partitioning. We expect SDSS_J queries to benefit less because their dominant operators are the joins, which cannot be improved by partitioning. SDSS_S queries on the other hand, benefit significantly since they are bound by I/O.

Figure 4 (a) shows the estimated workload performance for the two query groups. Partitioning improves SDSS_S queries by 2 or 3 times, even without the use of any indexes. In addition, column replication is beneficial for performance, with the schema using replication (AP1.5x) performing 10%better than the schema without (AP1x). SDSS_J queries improve much less (19% to 24%), since partitioning does not benefit join performance.

6.3 Indexing a Partitioned Schema

In this section we extend our comparison by building indexes on the partitioned and the original schemas. We design indexes using the Index Tuning Wizard in SQL Server 2000. We allow unlimited storage for indexes, but add updates (SDSS_U) to the input workload, in order to compare the efficiency of the designs. We also consider a set of hand-tuned indexes, designed by the experts maintaining the SDSS database, which were included with our SDSS distribution.

Figure 4 (b) shows the total workload cost when using the indexed original (I_ORIG, I_MAN) and partitioned schemas (I_AP1.5x and I_AP1x) for all statement groups. I_ORIG is the indexed schema generated by the Index Tuning Wizard and I_MAN is the manual design. Due to the effectiveness of partitioning the partitioned schemas use fewer and smaller indexes. Because of the reduced. indexing requirements, workload (SDSS_U) shows a dramatic 5x improvement for the partitioned schemas. Despite the fewer indexes, queries in the SDSS_S class improve by 20% compared to I_ORIG, because of the combined effect of partitioning and indexing. The performance improvement is similar for the replication and the no-replication case, because indexes mitigate any performance differences. Overall, partitioning improves query execution performance even in the presence of indexes, by 45% compared to I_ORIG.

Our experiment highlights the effectiveness of automated design techniques, since I_MAN is shown to perform worse than I_ORIG for the query workload (although it does better for updates). In addition, it confirms the benefits of automated partitioning as a physical design technique: We find that our partitioned-only schema

(AP1.5x), Figure 4 (a)) has similar performance to the I_MAN indexed design, without the overhead!

Figure 4 (c) details the total storage allocated for the I_ORIG, I_MAN and the two partitioned schemas, broken down into the storage required to index the two main tables. The 'covering' indexes built on the non-partitioned schemas require additional space larger than the original dataset size. The partitioned schemas require about half the space. Both I_ORIG and I_MAN rely on heavily indexing *PHOTOOBJ* for performance. In comparison, because of the performance benefits of partitioning *PHOTOOBJ*, the partitioned schemas require minimal storage to index *PHOTOOBJ* (4 and 7 times less than I_ORIG). Instead of heavily indexing *PHOTOOBJ*, the partitioned schemas allocate some more space for the efficient indexing of *NEIGHBORS*.

7 Summary

Database applications that use multi-terabyte datasets are becoming increasingly important for scientific fields such as astronomy and biology. In such environments, physical database design is a challenge. We propose AutoPart, an algorithm that automatically partitions database tables utilizing prior knowledge of a representative workload. AutoPart suggests an alternative, high-performance schema that executes queries faster than the original one and can be indexed using a fraction of the space required for indexing the original schema. To evaluate AutoPart, we build an automated schema design tool that interfaces to commercial database systems. This paper describes our algorithm, the system architecture, and experimental results using the Sloan Digital Sky Survey database.

References

[1] Sanjay Agrawal, Surajit Chaudhuri, and Vivek R. Narasayya. Automated selection of materialized views and indexes in SQL databases. *VLDB* (2000), 496–505.

[2] S. Ceri, M. Negri, and G. Pelagatti. Horizontal data partitioning in database design. *SIGMOD* (1982), 128–136.

[3] Damianos Chatziantoniou and Kenneth A. Ross. Groupwise processing of relational queries. *VLDB* (1997) 476–485.

[4] Surajit Chaudhuri and Vivek R. Narasayya. An efficient cost-driven index selection tool for Microsoft SQL server. *VLDB* (1997), 146–155.

[5] Douglas W. Cornell and Philip S. Yu. A vertical partitioning algorithm for relational databases. *ICDE* (1987) 30–35.

[6] G.Valentin, M.Zuliani, D.Zilio, and G.Lohman. DB2 advisor: An optimizer smart enough to recommend its own indexes. *ICDE* (2000), 101.

[7] Michael Hammer and Bahram Niamir. A heuristic approach to attribute partitioning. *SIGMOD* (1979), 93–101.

[8] Jeffrey A. Hoffer and Dennis G. Severance. The use of cluster analysis in physical data base design. *VLDB* (1975), 69–86.

[9] J.Gray, D.Slutz, A.Szalay, A.Thakar, J.vandenBerg, P.Kunszt, and C.Stoughton. Data mining the SDSS skyserver database. Technical Report MSR-TR-2002-01, Microsoft Research, 2002.

[10] Shamkant Navathe, Stefano Ceri, Gio Wiederhold, and Jinglie Dou. Vertical partitioning algorithms for database design. *ACM Trans. Database Syst.*, 9(4):680–710, 1984.

[11] Shamkant B. Navathe and Mingyoung Ra. Vertical partitioning for database design: a graphical algorithm. *SIGMOD* (1989), 440–450.

[12] Jun Rao, Chun Zhang, Nimrod Megiddo, and Guy Lohman. Automating physical database design in a parallel database. *SIGMOD* (2002), 558–569.

[13] S.Papadomanolakis and A.Ailamaki. AutoPart: Automated schema design for large scientific databases using data partitioning. *SSDM* (2004).

[14] Alexander S. Szalay, Jim Gray, Ani R. Thakar, Peter Z. Kunszt, Tanu Malik, Jordan Raddick, Christopher Stoughton, and Jan vandenBerg. The SDSS skyserver: public access to the sloan digital sky server data. *SIGMOD* (2002), 570–581.

[15] Toby J. Teorey and James P. Fry. The logical record access approach to database design. *ACM Comput. Surv.*, 12(2):179–211, 1980.

Logical and Physical Data Independence for Native Scientific Data Repositories

Bill Howe
OGI School of Science & Engineering at
Oregon Health & Science University
Beaverton, Oregon
bill@cse.ogi.edu

David Maier
Department of Computer Science
Portland State University
Portland, Oregon
maier@cs.pdx.edu

Abstract

Many datasets in the physical sciences, especially the results of simulations, are defined over a topological grid structure. Applications in these domains would benefit from a principled interface to gridded datasets via a specialized data model. Traditionally, benefits of a data model are realized only after data is ensconced within a managed database environment. However, massive bulk-loading and re-loading operations in large-scale data repositories are prohibitively expensive. Instead, we superimpose a specialized data model over native data repositories stored on directly on OS filesystems rather than managed by a database system. Views in a specialized data model can be defined via references to native directory structures and file content, providing physical and logical data independence. This non-intrusive approach appears to reduce space requirements, speed development, and cooperate with legacy applications.

1 Introduction

Scientific data repositories are approaching petabyte scales. Data manipulation in the context of such large repositories benefits from a well-defined data model interface. Physical and logical data independence provided by a data model insulate applications from the organization details and provide custom interfaces without copying and restructuring data. Algebraic optimization can recover performance penalties incurred by an extra layer of software and simplify previously hand-optimized applications.

In practice, most data models are implemented as heavyweight database systems requiring that legacy data be injected into a managed environment before benefits can be realized. Loading and re-loading data into database environments during development, and even simply inserting new data after successful deployment, can become infeasible. In our experience, bulk-loading simulation results into a relational database took almost as long as the simulation itself [4].

Further, legacy applications dependent on the existing interface are entrenched within the infrastructure. Maintaining two copies of data during the transition, one managed and one native, is a possibility since updates are rare, but storage space can be quickly depleted.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

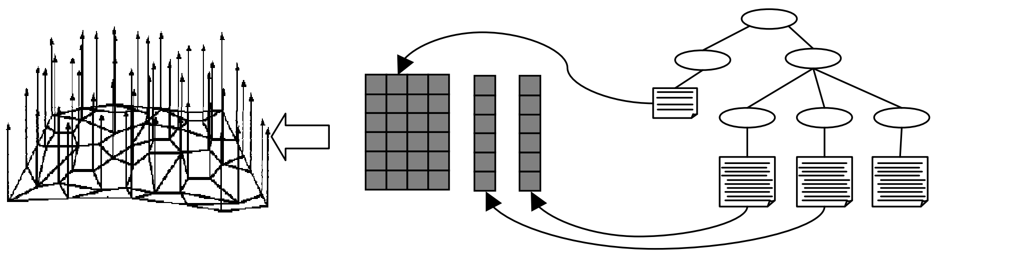


Figure 1: Mapping data from a filesystem model to an array model to a gridfield model.

Another approach, advocated in this paper, is to superimpose a data model over *in situ* data to provide principled access methods, specifically in Earth science domains. That is, given a scientific data repository stored directly on a filesystem, we aim to 1) describe it and 2) access it through a more specialized grid-based data model developed in previous work [4].

The context for our work is the CORIE Environmental Observation and Forecasting System being developed at the OGI School of Science and Engineering at Oregon Health and Science University [1]. The CORIE system is a multi-purpose platform for studying the fluid dynamics of the Columbia River estuary. Customers of CORIE's data products include commercial fisheries, environmental policy makers, and external research institutions. The CORIE repository consists of forecast and "hindcast" simulations covering time periods since 1998. Each day, forecast simulation runs add about 5GB to the data repository, while batches of hindcast runs, batches of calibration runs, and individual researchers' experiments are executed concurrently.

Simulation results are encoded in custom binary files and arranged in complex directory hierarchy. Three-dimensional spatial datasets are produced for each instant of simulated time, for each of several physical variables. These timestep datasets are distributed across several *checkpoint* files, each one usually covering a 24-hour period of simulated time. Checkpoint files have a custom binary format, and are arranged in a directory structure by week, by code version, and sometimes by purpose; e.g., calibration runs as opposed to final results. As a concrete example, a checkpoint file for the first Saturday of 2004 might have the path `hindcasts/01-2004/1_salt.63`. Every application accessing these data must understand the semantics of file and directory names, or interpret custom binary file formats, or both. The resulting situation is that much of the CORIE software is rather brittle with respect to changing data characteristics.

In previous work, we proposed a *gridfield* data model for manipulating gridded datasets [4] independently of physical organization details. The gridfield data model promotes topological structures to first-class entities, exposing equivalences and enabling optimization opportunities hidden by other tools. For example, the expression `Restrict(a>b, Cross(A, B))` computes a kind of join between two gridfields A and B, while preserving certain topological invariants.

In order to provide a gridfield interface to CORIE's significant existing data repository without destroying their native representations, we map the existing data structures (directories, files, arrays) into gridfields using *gridfield schema files*. Gridfield schema files describe how to translate source data into arrays, and thence how to assemble arrays to produce gridfields. Figure 1 illustrates the situation.

A gridfield schema file (henceforth *schema file*) holds a set of declarations. *FileType* declarations describe directory structures and allow access to data encoded within file and directory names. *BinaryBlocks* describe the layout of binary files. GridField declarations specify their constituent data through references to FileTypes and BinaryBlocks. With these declarations, we can assemble gridfields by pulling components together from multiple files, by extracting a portion of a single file, by extracting information from file names and directory names, or by an arbitrary combination of these methods. The finished schema file can then be *applied* to a *context* to determine which GridField declarations (if any) can be activated given the available data. With this mechanism, users have a common interface for both a structured data repository as well as individual files resulting from private experiments.

Effective use of information encoded in file and directory names requires more than simply invoking OS system calls and parsing strings. The logical structure of a data repository is frequently reflected in the directory structure, and filenames carry semantic information about their relationships with other files. A thorough model should allow transparent access to information in both a file's name and its content.

To illustrate, consider that the boundary between file name and file content can change depending on the situation. For example, the directory tree illustrated in Figure 3a has a separate file for each day of the week (for each variable). In this case, the day of the week is not stored within the file, and is therefore inaccessible to tools for reasoning only about a file's content [5, 7, 8]. This representation reflects the manner in which the data was generated: A checkpoint file was recorded for each day of the week to simplify recovery in case of failure. An individual researcher's ad hoc experiment might not require such caution; she might lump a week's worth of results into a single file without saving any checkpoints. In order to provide transparent access to either of these two representations, the model must capture information that spans file and directory names as well as file contents.

We acknowledge two roles involved with this framework. The *Schema Designer* writes FileType, BinaryBlock, and Gridfield declarations to a text file and publishes it to a well known location. The *Application Developer* applies the schema file to a context and manipulates available gridfields by writing expressions in a gridfield algebra. To preserve physical data independence, application developers are not expected to work directly with FileTypes or BinaryBlocks.

We do not support updates through the gridfield interface, though we do support some forms of append operations. Scientific data repositories tend to be append-only due to the need for provenance; even "bad" data tends to be kept, uncorrected, if space permits. Though support for updates to data values is not a priority, minimizing the impact of evolving representations is. By allowing arbitrary gridfield expressions to appear in a schema file, we provide a simple view facility and therefore logical data independence. Applications using the gridfield interface are insulated from changing representations. Only those portions of the schema file affected by the change must be adjusted by the schema designer.

The following list summarizes the benefits of a our approach.

- **Transparent interoperation with legacy applications.** No reorganization of the data is required for use.
- **Incremental return on investment.** As additional schema declarations are recorded, additional functionality is supported.
- **Transparent support for partially conforming data.** By not requiring all declarations to be activated in all contexts, the same schema file can be used as an interface to private data (with less structure perhaps) and public data (with more structure).
- **Generality.** External sources of data published to the web often follow a filesystem-like data model, and can therefore be accessed with the same framework as local data repositories.

2 GridField Data Model

A gridfield consists of a *grid*, and one or more *attributes*. A grid is a set of *cells*, partitioned by dimension. Each cell of dimension $d > 0$ (d-cell) is defined by a sequence of references to cells of dimension $d = 0$ (*nodes*). For example, a triangle is a sequence of three references to nodes. Every non-empty grid must have at last one 0-cell. Each attribute is bound to the cells of exactly one dimension d, such that each d-cell maps to exactly one value of the attribute.

a)

```
1: GridField G
2: G.grid.cells[0] = implicit 20
3: G.grid.cells[d] = <cell array expr>
4: G.a[d] = <array expr>
```

b)

```
1: GridField Horiz
2: Horiz.grid.cells[0] = implicit 10
3: GridField Vert
4: Vert.grid.cells[0] = implicit 15

5: GridField G = Cross(Horiz, Vert)
6: G.temperature(0) = <array expr>
```

Figure 2: Two examples of gridfield assembly syntax.

In this section we discuss how to assemble gridfields from arrays. An array named $name$ with n elements of type τ will be written `name[n]` (τ). Elements may be of a primitive numeric type, a tuple of named attributes, or another array. We will describe the source data model of arrays and tuples in more detail in the Section 4.

An example of a GridField declaration appears in Figure 2a. The nodes for the grid component of the gridfield G can be defined by an array expression returning integers (explicit node references) or by using the keyword `implicit`. The declaration in Figure 2 specifies that the grid of G will have 20 nodes when assembled. The third line in Figure 2a specifies how to construct a set of cells of a dimension $d > 0$ (e.g., line segments, triangles, tetrahedra). The placeholder <cell array expr> must have one of the following two forms.

$$array[N](m : int, array[m](n_1 : int, n_2 : int, ..., n_m : int))$$

where N is the number of cells (the size of the array), and each cell is a sequence of m references to a node. The second form is

$$array[N](n_1 : int, n_2 : int, ..., n_M : int))$$

where N is the number of cells and every cell has M nodes. The first form corresponds to mixed cell types found in grids used for finite-element simulation (e.g., both triangles and quadrilaterals appear in the same grid.) The second form corresponds to the uniform cell types found in graphics applications (e.g., triangles in two dimensions or tetrahedra in three).

To bind an attribute a to cells of dimension d, we use the syntax in line 4 of Figure 2a. The placeholder <array expr> must return an array of primitive elements (no nested sequences or arrays). Array elements are associated with cells positionally; the first cell is bound to the first array element, the second cell is bound to the second array element, and so on.

We can also remove the integer argument to the keyword `implicit`. Without this argument, the keyword indicates that the number of nodes is unconstrained, and may be derived from the number of values in an attribute bound to the 0-cells. When the argument is present, the number of nodes is constrained to be the argument's value, and binding attributes with a different cardinality results in an error.

Gridfields can also be declared through expressions of a gridfield algebra, as in Figure 2b. We first define two gridfields `Horiz` and `Vert` in lines 1 and 3, and then construct their *cross product* in line 5. The grid `G` will have 150 nodes and one bound attribute, `temperature`. The details of this and other gridfield operators can be found in a previous paper [4].

In the next section, we will describe how to extract the requisite arrays from native data formats.

3 Modeling Filesystem Data

Scientists frequently store and manage their data using direct filesystem interfaces. Limitations to this approach are evident. A file's metadata must frequently be embedded in file and directory names. For ex-

a)
```
/run /01-2004 /1_salt.63
              /1_temp.63
              /2_salt.63
              /2_temp.63
              :
     /02-2004 /1_salt.63
              /1_temp.63
              /2_salt.63
              /2_temp.63
              :
```

b)
```
FileType weekly_run
pattern[wk,yr] = /run/%i-%i/
FileType salt63
pattern[day] = %i_salt.63
FileType temp63
pattern[day] = %i_temp.63

GridField Days
G.grid.cells[0] = implicit
G.day[0] = salt63.day
```

c)
```
FileType alldays
pattern[wk,yr,day]
   = /run/%i-%i/%i_salt.63

GridField AllDays
G.grid.cells[0] = implicit
G.week[0] = alldays.wk
G.year[0] = alldays.yr
G.day[0] = alldays.day
```

Figure 3: A gridfield schema for extracting information from a filesystem directory structure.

ample, a dataset available from the National Climate Data Center (NCDC) website is stored in a file named `meso-eta_215_20030803_1800_fff` [6]. The meaning of the date string "20030803" is apparent, but the other fields require some external information to parse. Constraints are also difficult to express. The file from the NCDC above was found in a directory named `20030803`. Presumably, all files within this directory should be tagged with this date, and those that are not represent anomalies.

The example in Figure 3b is taken from the CORIE system. The root directory `runs` contains directories with simulation results for each week of 2004 in the form `<week number>-2004`. Each week directory contains 14 files, one for each variable (salinity, temperature) and day of the week (1-7). To access the week number or day number embedded in these file names, we can write a *path pattern* in the style of the Unix scanf command.

```
[week, year, day] = /runs/%i-%i/%i_salt.63
```

The left-hand side of this expression is a tuple of variable names. The right-hand side is a pattern matched against the set of all files in some filesystem context. Wildcard placeholders are given a one-character type code (`i` for integer, `f` for float, etc.). Each variable name can be accessed as a sequence of values once the pattern is applied to a context. Note that the sequence order is imposed by the directory traversal; a potential extension to the language is to allow arrays to be sorted to enforce a different ordering.

Path patterns are lifted into FileTypes by assigning them a type name. Given a filesystem context, a FileType can be evaluated to return a sequence of tuples whose attribute names are given by the list of variable names, and whose types are given by the type codes of the wildcards. For example, we can define separate FileTypes for the run directories, the salinity data, and the temperature data as in Figure 3b.

A simple GridField declaration referencing these FileTypes appears at the bottom of Figure 3b. The gridfield `Days` consists of 0-cells only, and its cardinality is determined implicitly by the number of files that match the path pattern. The array expression for attribute `day` of the declaration references the FileType `salt63`, and then uses the dot notation to access one of its sequence variables. If the schema in Figure 3b were applied to the directory `02-2004/` of Figure 3a, the gridfield `Days` would consist of seven nodes, since there are seven files that match the pattern. Note that we may chose `temp63` to define the `Days` gridfield. The ambiguity reflects the fact that the `day` attribute is redundantly encoded in the filenames of all simulation variables.

Another set of GridField declarations appears in Figure 3c. The gridfield `AllDays` effectively "unnests" the day numbers from within the week directories, producing (week, year, day) tuples.

4 Modeling File Content

Scientists frequently use packed binary encodings of large datasets to preserve space and improve performance. We model the content of binary files using named primitive types, records (where each element is named and

a)
```
BinaryBlock image
image.content = (
  header : 10c
  h : i
  w : i
  data : h*(
    row : w*(
      r:i g:i b:i
    )
  )
)
```

b)
```
GridField Horizontal
Horizontal.grid.cells[0] = implicit image.h

GridField Vertical
Vertical.grid.cells[0] = implicit image.v

GridField Image = Cross(Horizontal, Vertical)
Image.r[0] = image.data.r
Image.r[0] = image.data.g
Image.r[0] = image.data.b
```

Figure 4: A gridfield schema file for extracting binary file content.

may have a different type) and arrays (where all elements have the same type and are referenced by position only). Both structures support arbitrary nesting. Arrays are one-dimensional; multidimensionality is captured through nesting.

A schema for a block of binary content representing an image appears in Figure 4a. The top-level component is named `image`, which contains a 10-character string named `header`, the height `h` and width `w` of the image, and a two-dimensional array of RGB values. The height and width are primitive components of the form `name : type`, where type in this case is `i` signifying an integer. The data component has a type of the form `x*(component)` where x is the name of an integer component (or is an integer literal) and *component* is another named component in the current scope. In this case, another array component is nested within the first to represent the second dimension.

Tuple elements are accessed by name or position. An array element is accessed through conventional integer indexing. An array can also be "sliced" to produce another array as in APL or Matlab. Some examples of these expressions appear in Table 2.

This language is not particularly expressive: For example, we cannot transpose, sort, or aggregate arrays. Array expressions are meant only as a means of assembling gridfields from raw binary data; we rely on gridfield algebra expressions to perform more complex transformations.

To construct gridfields from BinaryBlocks, we use a dot notation syntax similar to that used in Section 3, as in Figure 4b. We discuss combining FileTypes and BinaryBlocks in Section 5.

5 Nested Gridfields

Each GridField declaration is evaluated against a *context*. A context is either the root context, a FileType, a BinaryType, or a concatenation of contexts. The root context is provided by the application developer during connection to the data repository. So far, we have written declarations as if they all are evaluated against this root context. To assemble more complex *nested* gridfields, we need to pass a context in as an argument to each declaration. Effectively, GridField declarations are functions mapping contexts to gridfields.

Consider the schema in Figure 5, where the GridField declarations are adorned with a context argument. As an argument, a context represents a single entity. Using the dot notation, we can concatenate contexts and access data items within contexts. For example, for the GridField declaration `Time`, `ROOT` is a single entity, the directory (or file) supplied by the application developer to which the schema is being applied. The expression `ROOT.gridfile.time`, however, will return an array of values extracted from the time field of FileType `gridfile`. The expression `ROOT.gridfile.scalardata` requests that the contents of each grid file be interpreted according to the binary block declaration `scalardata`. Since each element of this array has complex structure, we cannot construct an ordinary attribute of primitive values. However, we can construct a

Table 2: Example expressions and their types.

expression	type
`image.h`	`integer`
`image.data.row[1]`	`(R:i, G:i, B:i)`
`image.data.row.R`	`array[h*w](R:i)`
`image.data[a:b]`	`array[b-a](array[w](R:i, G:i, B:i))`
`image.data.row[a:b:c]`	`array[(b-a)/c](array[w](R:i, G:i, B:i))`

nested gridfield, where each element of a new attribute is itself a gridfield.

The syntax for a nested gridfield involves referencing the name of the GridField declaration and passing it an array expression as an argument. Each element of the array expression is considered a context in which to assemble a gridfield. Therefore, the syntax is effectively shorthand for a list comprehension expression, as denoted by the C-style comment below the declaration of GridField `Time`.

In the declaration for GridField `Geometry`, a grid with explicit triangular cells is constructed. The gridfield has x and y values bound to the nodes representing the position of each node in two-dimensional space. The gridfield `Temp` and `Velo` are both assembled from the gridfield `Geometry`. Of course, we might have decided to model these data as a single gridfield with three attributes `temp`, `u`, and `v`. By separating the temperature data from the velocity data ,we illustrate the variety of modeling choices open to the schema designer.

6 Related Work and Discussion

Scientific applications today in some ways resemble business applications circa 1977. Copious amounts of data are stored in files with intricate formats. Skepticism regarding database technology is rampant. Legacy systems are built from efficient but extremely brittle software components. To mitigate the perceived (and real) risk of adopting unproven database systems, early data models were implemented as file transformation engines.

The EXPRESS system [7] provided two languages: one for describing a file's structure, and another for transforming that structure. Transformations were used as a query facility, but also as a bulk load facility to translate legacy data into a new format. Our approach is similar, though we distinguish two data models: one for source data (directory structures and file content) and another for target data (gridfields). We have not yet considered materializing gridfields assembled using schema files. That is, we do not permanently transform source data into gridfields, but rather provide a gridfield interface to native data.

Batory gave a taxonomy of record-oriented file structures used by commercial databases in terms of fields and pointers [2]. Our work similarly provides a description of grid-oriented file structures in terms of arrays. Once precisely described, Batory showed how you could convert from one representation to another in a principled manner, or compose the transforms. Tools to convert between array-oriented representations are valuable, though we are primarily interested in lifting arrays into more semantic structures, particularly gridfields.

The Binary Format Description Language (BFD) [5] is an XML dialect that describes binary formats and allows transformation of binary data to XML data. While this tool has a niche, our interest is to support efficient and flexible access to binary data – converting binary data to XML is clearly impractical for large datasets. The BinX [8] library is also related to this proposal. Binary data file formats are described using instances of a specialized XML Schema. An API allows access to the data and automatic reformatting according to the local machine's byte order and bit order. The current proposal aims to provide more powerful reasoning capabilities than simple file access. BinX supports only sequential file access, and file structures are all assumed to fit in memory. BinX also offers no support for transforming representations; any restructuring is done "manually" in a C program after reading in an entire source file.

Schema integration techniques, particularly the concept of local-as-view [3], is also related to our approach.

```
FileType gridfile
pattern[time] = %i_*.grd

BinaryBlock griddata
content = (
  nodes : i
  elements : i
  triangles : elements*
    ( n1:i n2:i n3:i )
  geometry : nodes*
    ( x:i y:i )
  velocity : elements*
    ( u:f v:f )
  temp : elements*
    ( value:f )
)
```

```
GridField Geometry(griddata)
Geometry.cells[0] = implicit griddata.nodes
Geometry.cells[2] = griddata.triangles
Geometry.x[0] = griddata.geometry.x
Geometry.y[0] = griddata.geometry.y

GridField Temp(griddata) = Geometry(griddata)
Temp.temp[2] = griddata.temp.value

GridField Velo(griddata) = Geometry(griddata)
Velo.u[2] = griddata.velocity.u
Velo.v[2] = griddata.velocity.v

GridField Time(ROOT)
Time.cells[0] = implicit
Time.time[0] = ROOT.gridfile.time
Time.velo[0] = Velo(ROOT.gridfile.griddata)
Time.temp[0] = Temp(ROOT.gridfile.griddata)
// = [Temp(c) | c <- ROOT.gridfile.griddata]
```

Figure 5: A gridfield schema involving nested gridfields.

However, the source data being integrated (usually stored in relational databases) are more suitable for query and manipulation than our raw native repositories. Wrappers and mediators also serve to provide access to native data. However, data sources are usually modeled as black-boxes with a limited interface. Our context allows us to reason about the source data in considerable detail, though we cannot prescribe changes in its organization.

In conclusion, we advocate in situ processing of large scientific data repositories as a means of harnessing database techniques without the substantial cost of database deployment. Database technology is not prolific in the scientific community; the real and perceived cost of ownership is part of the reason. If some database techniques can be teased apart from their monolithic implementations, the database community stands a chance of contributing a tangible benefit to the scientific community.

References

[1] A. Baptista, M. Wilkin, P. Pearson, P. Turner, M. C., and P. Barrett. Coastal and estuarine forecast systems: A multi-purpose infrastructure for the columbia river. *Earth System Monitor, NOAA*, 9(3), 1999.

[2] D. S. Batory. Modeling the storage architectures of commercial database systems. *ACM Trans. Database Syst.*, 10(4):463–528, 1985.

[3] M. Friedman, A. Y. Levy, and T. D. Millstein. Navigational plans for data integration. In *AAAI/IAAI*, pages 67–73, 1999.

[4] B. Howe and D. Maier. Algebraic manipulation of scientific datasets. In *Proceedings of the 30th International Conference on Very Large Databases (VLDB2004)*, 2004.

[5] J. Myers and A. Chappell. Binary format description language. Technical report, Pacific Northwest National Laboratory.

[6] National Climatic Data Center. NCEP AWIPS eta model data. http://nomads.ncdc.noaa.gov:9090/dods/NCDC_NOAA Port_ETA.

[7] N. C. Shu, B. C. Housel, R. W. Taylor, S. P. Ghosh, and V. Y. Lum. EXPRESS: A data EXtraction, Processing, amd REStructuring System. *ACM Transactions on Database Systems*, 2(2):134–174, 1977.

[8] M. Westhead and M. Bull. Representing scientific data on the Grid with BinX - binary XML description language. Technical report, EPCC, University of Edinburgh, 2003.

Similarity Search in Biological and Engineering Databases

Hans-Peter Kriegel, Martin Pfeifle, Stefan Schönauer
Institute for Informatics
University of Munich
{kriegel, pfeifle, schoenauer}@dbs.ifi.lmu.de

Abstract

Similarity search is an important task in a wide range of scientific database applications. Besides being used directly, it is also used as a basic operation for many data mining algorithms. Example applications which are presented in this article are functional classification of proteins in biological databases and the similarity search of CAD parts in engineering environments. It turns out that the general concept of set-based similarity measures can be successfully applied to both of these diverse application domains.

1 Introduction

In the last ten years, an increasing number of database applications has emerged for which efficient and effective support for similarity search is substantial. The importance of similarity search grows in application areas such as multimedia, medical imaging, molecular biology, computer aided engineering, marketing and purchasing assistance, etc. [Jag91, MG93, Fal+94, ALSS95, BKK97, Kei99]. Particularly, the task of finding similar shapes in 2D and 3D becomes more and more important. Examples for new applications that require the retrieval of similar 3D objects include databases for molecular biology, medical imaging and computer aided design.

In recent years, considerable work on similarity search in database systems has been published. Many of the previous approaches, however, deal with 1D or 2D data, such as time series, digital images or polygonal data. In the following, we will shortly list different approaches to establish similarity measures known from literature. We provide a classification of the techniques into feature-based models and direct geometric models.

Feature-Based Similarity. A widely used class of similarity models is based on the paradigm of feature vectors. The basic idea is as follows: Using a feature transform, the objects are mapped onto a feature vector in an appropriate multidimensional feature space. The similarity of two objects is then defined as the proximity of their feature vectors in the feature space: The closer their feature vectors are located, the more similar two objects are considered.

The paradigm of feature-based similarity has been successfully applied to the retrieval of similar spatial objects. Examples include structural features of 2D contours [MG93], angular profiles of polygons [BMH92], rectangular covers of shapes [Jag91], algebraic moment invariants [Fal+94], 2D section coding [BKK97], and 3D shape histograms for biomolecular objects [AKKS99]. Non-geometric applications include similarity search on time series [ALSS95], and on color histograms in image databases [Fal+94], among several others.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

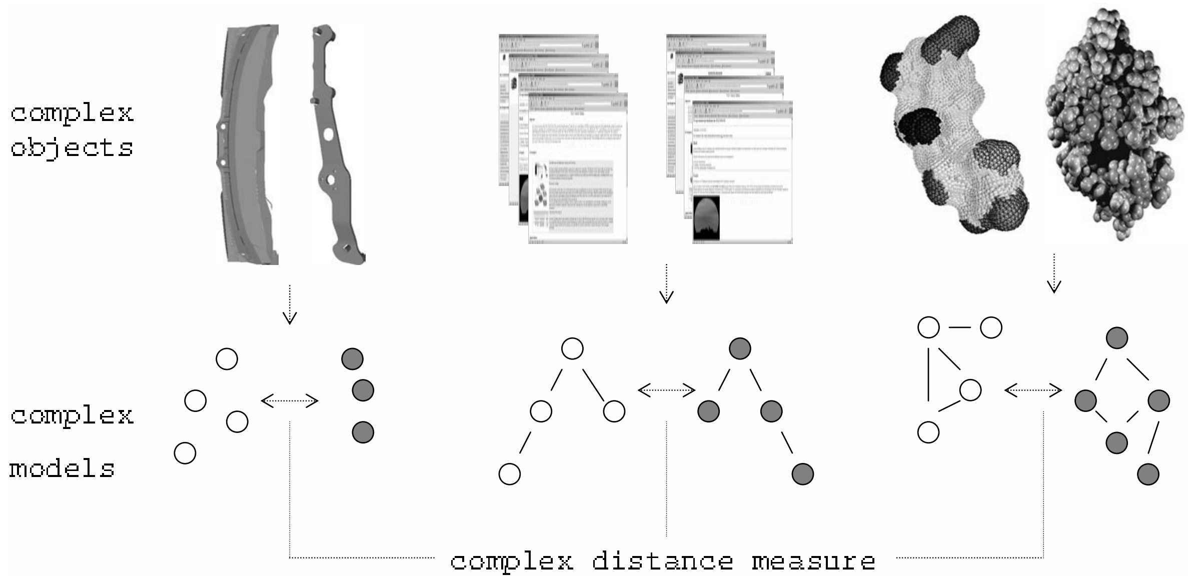


Figure 1: Examples of Complex Objects.

Geometry-Based Similarity. A class of models that is to be distinguished from the feature-based techniques are the similarity models that are defined by directly using the geometry of the objects. Two objects are considered similar if they minimize a distance criterion that is purely defined by the geometry of the objects. Examples include the similarity retrieval of mechanical parts [SKSH89], the difference volume approach [Kei99], and the approximation-based similarity model for 3D surface segments [KSS97].

Model-Based Similarity. In this paper, we propose to model complex real world objects by complex models (cf. Figure 1). As the development of conventional database systems in the recent two decades has shown, the use of more sophisticated ways to model data can enhance both the effectiveness and efficiency for applications using large amounts of data. In this paper, we will use point sets to describe CAD objects and graphs are used to represent protein molecules. The distance measure is in both cases based on a minimum weight perfect matching. Furthermore, we will sketch suitable filters for this rather complex distance function to speed up similarity query processing.

The remainder of the paper is organized as follows. In Section 2, we present a similarity model for voxelized CAD objects which are represented by sets of feature vectors. In Section 3, we present a similarity model for proteins which are modeled as graphs. The distance function used for measuring the similarity between two graphs, is based on the distance function used for measuring the similarity between two CAD objects. We close this paper in Section 4 with a short summary and a few remarks on future work.

2 Engineering Databases

The development, design, manufacturing and maintenance of modern engineering products is a very expensive and complex task. Effective similarity models are required for two- and three-dimensional CAD applications to cope with rapidly growing amounts of data. Shorter product cycles and a greater diversity of models are becoming decisive competitive factors in the hard-fought automobile and aircraft markets. These demands can only be met if the engineers have an overview of already existing CAD parts. In Section 2.2, we present a similarity model for complex 3D CAD data, which helps to find and group similar parts. This model is based on the cover sequence model which is introduced in the following section.

2.1 The Cover Sequence Model

In this section, we adapt a known model [Jag91, JB91] to voxelized 3D data, the so-called *cover sequence model*. The basic idea of this model is to find large clusters of grid primitives, called *covers*, which approximate the object in a best possible way [JB91].

The quality of such a cover sequence S_k is measured by the symmetric volume difference Err_k between the object O and the sequence S_k. Formally, let the covers be drawn from the set $\mathcal{C}$ of all possible rectangular covers. Then each unit i of the cover sequence comprises a pair $(C_i \in \mathcal{C}, \sigma_i \in \{+,-\})$, where "+" represents set union and "−" represents set difference. The sequence after k units is:

$$S_k = (((C_0 \sigma_1 C_1) \sigma_2 C_2) \ldots \sigma_k C_k),$$

where C_0 is an initially empty cover at the origin.

The symmetric volume difference after k units is:

$$Err_k = \mid O \text{ } \mathit{XOR} \text{ } S_k \mid \text{, where } O \text{ is the approximated object.}$$

In [Jag91], Jagadish sketches how a 3D cover sequence $S_k = (((C_0\sigma_1 C_1)\sigma_2 C_2)\ldots\sigma_k C_k)$ of an object O, can be transformed into a $6 \cdot k$-dimensional feature vector. Thereby, each cover C_{i+1} with $0 \leq i \leq k-1$ is mapped onto 6 values in the feature vector f_o in the following way:

$$\begin{aligned}
f_o^{6i+1} &= x\text{-position of } C_{i+1} \\
f_o^{6i+2} &= y\text{-position of } C_{i+1} \\
f_o^{6i+3} &= z\text{-position of } C_{i+1} \\
f_o^{6i+4} &= x\text{-extension of } C_{i+1} \\
f_o^{6i+5} &= y\text{-extension of } C_{i+1} \\
f_o^{6i+6} &= z\text{-extension of } C_{i+1}
\end{aligned}$$

2.2 Using Sets of Feature Vectors for Similarity Queries

As proposed in [Jag91] a data object is now represented as a feature vector. For similarity queries this method yields a major problem. Always comparing the two covers having the same ranking according to the symmetric volume difference, does not make sense in all cases. Two objects can be considered very different, because of the order of their covers, although they are very similar by intuition. The reason for this effect is that the order of the covers does not guarantee that the most similar covers due to size and position will be stored in the same dimensions. Especially for objects generating two or more covers having almost the same volume, the intuitive notion of similarity can be seriously disturbed. Thus, the possibility to match the covers of two compared objects using more degrees of freedom, might offer a better similarity measure.

The representation of extracted features as a set of vectors, i.e. each cover is represented by a 6-dimensional feature vector, is a generalization of the use of just one large feature vector. It is always possible to restrict the model to a feature space, in which a data object will be completely represented by just one feature vector. But in some applications the possibilities of vector set representation allow us to model the dependencies between the extracted features more precisely. In our application the vector set representation is able to avoid the problems that occur by storing a set of covers according to a strict order. Therefore, it is possible to compare two objects more intuitively, causing a relatively small rise of calculation cost compared to the cost of distance calculations in the one-vector model.

2.2.1 Distance Measures on Vector Sets

A distance measure on vector sets that demonstrates to be suitable for defining similarity in our application is based on the *minimum weight perfect matching* of sets. This well known graph problem can be applied here, by building a complete bipartite graph $G = (S_1 \cup S_2, E)$ between the vector sets S_1 and S_2. The weight of each edge $(x, y) \in E$ with $x \in S_1$ and $y \in S_2$ in this graph G is defined by their distance $dist(x, y)$. A perfect matching is a subset $M \subseteq E$ that connects each $x \in S_1$ to exactly one $y \in S_2$ and vice versa. A minimum weight perfect matching is a matching with a minimum sum of weights of its edges. Since a perfect match can only be found for sets of equal cardinality, it is necessary to introduce weights for unmatched nodes when defining a distance measure.

Definition 1: (enumeration of a set)
Let S be any finite set of arbitrary elements. Then π is a mapping that assigns $s \in S$ a unique number $i \in \{1, .., |S|\}$. This is written as $\pi(S) = (s_1, .., s_{|S|})$. The set of all possible enumerations of S is named $\Pi(S)$.

Definition 2: (minimal matching distance)
Let O be the domain of the objects and X be a set with $|X| \leq k$ and $X \subseteq 2^V$ where $V \subset I\!R^d$. Furthermore, let $F : O \to X$ be a mapping of the objects into X, and *dist* : $I\!R^d \times I\!R^d \to I\!R$ a distance function between two d-dimensional feature vectors. We assume w.l.o.g. $|F(Obj_1)| = m \geq n = |F(Obj_2)|$, $F(Obj_1) = \{x_1, .., x_m\}$ and $F(Obj_2) = \{y_1, .., y_n\}$. Then $dist_{mm}^{w,dist}$: $O \times O \to I\!R$ is defined as follows:

$$dist_{mm}^{w,dist}(Obj_1, Obj_2) = \min_{\pi \in \Pi(F(Obj_1))} \left(\sum_{i=1}^{n} dist(x_{\pi(i)}, y_i) + \sum_{l=n+1}^{m} w(x_{\pi(l)}) \right)$$

where $w : I\!R^d \to I\!R^+$ is a weight function for the unmatched elements.

The weight function w provides the penalty given to every unassigned element of the set having larger cardinality. Let us note that *minimum matching distance* is a specialization of *netflow distance* which is introduced in [RB00]. In [RB00] it is proven that netflow distance is a metric and that it is computable in polynomial time. Though it was shown that the netflow distance can be calculated in polynomial time, it is not obvious how to achieve it. Since we are only interested in the minimum matching distance, it is enough to calculate a minimum weight perfect matching. Therefore, we apply the method proposed by Kuhn [Kuh55] and Munkres [Mun57]. The method is based on the successive augmentation of an alternating path between both sets. Since it is guaranteed that this path can be expanded by one further match within each step taking $O(k^2)$ time and since there is a maximum of k steps, the all over complexity of a distance calculation using the method of Kuhn and Munkres is $O(k^3)$ in the worst case. Let us note that for larger numbers of k this is far better than the previously mentioned method on $k!$ many permutations.

For more details about the presented similarity model based on sets of feature vectors, we refer the interested reader to [BKK+03]. In [BKK+03], there is also a $centroid$ filter presented which helps to accelerate similarity queries on vector set represented objects. The basic idea is that for each vector set we compute its centroid. The difference between the two centroids of the two point sets times the cardinality of the point sets forms a lower-bound for the minimal matching distance.

3 Biological Databases

Some of the concepts developed for engineering databases can also be used in biological databases. Exemplarily, we describe the application in the field of protein function prediction.

3.1 Analysis and Prediction of Protein Function

For the analysis and prediction of the interaction of proteins, the molecular surface is of particular interest. The 3D geometry of molecules is a highly selective criterion for the functional interaction of proteins. Advanced experimental methods for structure determination and the application of algorithms to predict protein folds result in tens of thousands of reliably known molecular conformations. Faced with these increasing data volumes, the introduction of database techniques to manage the data becomes more and more vital. One important task in this context is the functional classification of proteins based on homology search. This task can be accomplished using knn-classifiers for the functionally active 3D-surface of protein molecules.

To represent such complex objects as biologically active molecules, rich models integrating geometrical, structural and biochemical information are needed. Therefore, we employ the concept of feature graphs as a model for protein surfaces, which are a type of attributed graphs. They not only allow for the integration of biochemical information, but also offer the possibility to consider molecular flexibility, a key feature in this application domain. This is the basis for an effective functional classification of protein molecules.

3.2 Similarity of Graphs

As graphs are a very general object model, graph similarity has been studied in many fields. Similarity measures for graphs have been used in systems for shape retrieval [HCH99], object recognition [KKV90] or face recognition [WFKM97].

A very common similarity measure for graphs is the edit distance. It uses the same principle as the well known edit distance for strings [Lev66, WF74]. The idea is to determine the minimal number of insertion and deletions of vertices and edges to make the compared graphs isomorphic. In [SF83], Sanfeliu and Fu extended this principle to attributed graphs, by introducing vertex relabeling as a third basic operation beside insertions and deletions. Unfortunately, the edit distance is a very time-complex measure. Zhang, Statman and Shasha proved in [ZSS92] that the edit distance is MAX–SNP–hard even for unordered labeled trees.

In the field of image retrieval, similarity of attributed graphs is sometimes described as an assignment problem [Pet02], where the similarity distance between two graphs is defined as the minimal cost for mapping the vertices of one graph to those of another graph. With an appropriate cost function for the assignment of vertices, this measure takes the vertex attributes into account and can be evaluated in polynomial time. This assignment measure, which we will call vertex matching distance in the following, obviously completely ignores the structure of the graphs, i.e. they are just treated as sets of vertices.

As we just described, all the known similarity measures for attributed graphs have certain drawbacks. Starting from the edit distance and the vertex matching distance we proposed a new method to measure the similarity of attributed graphs. This method solves the problems mentioned above and is useful in the context of large databases of structured objects. For our similarity measure, called the edge matching distance, we rely on the principle of graph matching, just like in the case of the vertex matching distance. But instead of matching the vertices of two graphs, we propose a cost function for the matching of edges and then derive a minimal weight maximal matching between the edge sets of two graphs. This way not only the attribute distribution, but also the structural relationships of the vertices are taken into account.

Definition 3 (edge matching distance): Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be two attributed graphs. Without loss of generality, we assume that $|E_1| \geq |E_2|$. The complete bipartite graph $G_{em}(V_{em} = E1 \cup E_2 \cup \Delta, E_1 \times (E_2 \cup \Delta))$, where Δ represents an empty dummy edge, is called the edge matching graph of G_1 and G_2. An edge matching between G_1 and G_2 is defined as a maximal matching in G_{em}. Let there be a non-negative metric cost function $c : E_1 \times (E_2 \cup \Delta) \rightarrow \mathbb{R}_0^+$. The edge matching distance between G_1 and G_2, denoted by $d_{match}(G_1, G_2)$, is defined as the cost of the minimum-weight edge matching between G_1 and G_2 with respect to the cost function c.

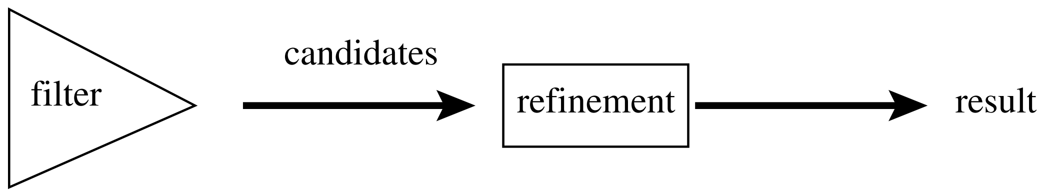


Figure 2: A filter-refinement architecture.

This measure is similar to the matching distance of definition 2. Especially, the algorithm of Kuhn [Kuh55] and Munkres[Mun57] can be used to determine the minimum-weight matching, which results in a time-complexity of $O(n^3)$ where n denotes the number of edges, i.e. $n = |E_1|$.

The edge matching distance has several properties, which are important for the application of the measure to similarity search and to classification systems. One is the fact that the edge matching distance is a metric. This allows the application of several methods to speed up query processing. Another such property is the polynomial time-complexity of the measure as opposed to the exponential time-complexity of the edit distance. The polynomial time-complexity is the precondition for the application of the measure in large databases, where the measure has to be calculated repeatedly even when a single query is processed.

As experiments show, the time-complexity of the edge matching distance is still too high for use in a classification system. Therefore we developed techniques for efficient query processing with the edge matching distance, based on the concept of multi-step query processing.

Query processing in a multi-step query processing architecture, as depicted in figure 2, is performed in two or more steps, where the first steps are filter steps that return a number of candidate objects from the database. For those candidate objects, the exact similarity distance is determined in the refinement step and the objects fulfilling the query predicate are reported. To reduce the overall search time, the filter steps have to be cheap to perform and a substantial part of the database objects has to be filtered out.

Additionally, the completeness of the filter step is essential, i.e. there must be no false drops during the filter steps. Available similarity search algorithms guarantee completeness if the distance function in the filter step fulfills the lower-bounding property. This means that the filter distance between two objects must always be less than or equal to their exact similarity distance.

Using a multi-step query processing architecture requires efficient algorithms which actually make use of the filter step. Agrawal, Faloutsos and Swami proposed such an algorithm for range search [AFS93]. In [SK98] and [KSF+98] multi-step algorithms for k-nearest-neighbor search were presented, which are optimal in the number of exact distance calculations necessary during query processing. Therefore, we use the latter algorithms in our classification system.

To employ a filter-refinement architecture we need filters for the edge matching distance, which cover the structural as well as the attribute properties of the graphs in order to be effective. A way to derive a filter for a similarity measure is to approximate the database objects and then determine the similarity of those approximations. As an approximation for the structure of a graph we use the size of that graph, i.e. the number of edges in the graph. A lower bound for the edge matching distance between two graphs can be derived from the difference between their sizes. Our filters for the attribute part of graphs are based on the observation that the difference between the attribute distributions of two graphs influences their edge matching distance. Obviously, it is too complex to determine the exact difference of the attribute distributions of two graphs in order to use this as a filter and, therefore, an approximation of those distributions is needed. We propose a filter for the attribute part of graphs, which exploits the fact that $\forall x, y \in \mathbb{R} : |x - y| \geq ||x| - |y||$. For attributes which are associated with edges, we add all the absolute values for an attribute in a graph. For two graphs G_1 and G_2 with equal size, the difference between those sums is the minimum total difference between G_1 and G_2 for the respective attribute. Weighted appropriately according to the cost function that is used, this is a lower bound for the edge matching distance. For graphs of different size, this is no longer true, as an edge causing the attribute difference could also be assigned to an empty edge. Therefore, the difference in size of the graphs multiplied with the maximum cost for this attribute has to be subtracted from the previously computed value, in order to be lower bounding in all cases.

When considering attributes that are associated with vertices in the graphs, we have to take into account that during the distance calculation a vertex v is compared with several vertices of the second graph, namely exactly $degree(v)$ many vertices. To take care of this effect, the absolute attribute value for a vertex attribute has to be multiplied with the degree of the vertex, which carries this attribute value, before the attribute values are added in the same manner as for edge attributes. Obviously, the appropriately weighted size difference has to be subtracted in order to achieve a lower bounding filter value for a node attribute.

With the above methods it is ensured that the sum of the structural filter distance plus all attribute filter distances is still a lower bound for the edge matching distance between two graphs. Furthermore, it is possible to precompute the structural and all attribute filter values and store them in a single vector. This supports efficient filtering during query processing.

The edge matching distance and the filtering methods are more thoroughly described in [KS03].

First experiments on the effectiveness and efficiency of our classification approach already show encouraging results. This underlines the applicability of general concepts developed for engineering databases in the area of biological databases.

4 Conclusions

Similarity search is an important task in a wide range of scientific database applications. Besides being used on its own, it is also a basic operation for many data mining applications. In this paper, we presented two such data mining applications from the areas of engineering and biology. With the *minimal matching distance* and the *edge matching distance* we demonstrated that the concept of a similarity measure between sets can be successfully applied in two very diverse application areas. Additionally, the presented similarity measure can be efficiently handled even in large databases using multi-step query processing architectures.

In our future work, we would like to extend the presented concepts to other application domains as well to new problems in engineering and bioinformatics.

References

[AFS93] Agrawal, R. and Faloutsos, C. and Swami, A. N.: *Efficient similarity search in sequence databases*. Proc. 4th Int. Conf. of Foundations of Data Organization and Algorithms (FODO), Chicago, Illinois, pp. 69–84, 1993.

[AKKS99] Ankerst, M. and Kastenmüller, G. and Kriegel, H.-P. and Seidl T.: *3D Shape Histograms for Similarity Search and Classification in Spatial Databases*. Proc. 6th Int. Symposium on Large Spatial Databases (SSD'99), Hong Kong, China, LNCS 1651, pp. 207–226, 1999.

[ALSS95] Agrawal, R. and Lin, K.-I. and Sawhney, H.S. and Shim, K.: *Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases*. Proc. 21th Int. Conf. on Very Large Databases (VLDB'95), pp. 490–501, 1995.

[BKK97] Berchtold, S. and Keim, D.A. and Kriegel, H.-P.: *Using Extended Feature Objects for Partial Similarity Retrieval*. VLDB Journal, 6(4):333–348, 1997.

[BKK+03] Brecheisen S. and Kriegel H.-P. and Kröger P. and Pfeifle M. and Schubert M.: *Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects*. Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA, pp. 587-598, 2003.

[BMH92] Badel, A. and Mornon, J.P. and Hazout, S.: *Searching for Geometric Molecular Shape Complementarity using Bidimensional Surface Profiles*. Journal of Molecular Graphics, 10: 205–211,1992.

[Fal+94] Faloutsos, C. and Barber, R. and Flickner, M. and Hafner, J. and others: *Efficient and Effective Querying by Image Content*. Journal of Intelligent Information Systems, 3:231–262, 1994.

[HCH99] Huet, B. and Cross, A. and Hancock, E.: *Shape Retrieval by Inexact Graph Matching*. Proc. IEEE International Conference on Multimedia Computing Systems, pp. 40–44, 1999.

[Jag91] Jagadish, H.V.: *A Retrieval Technique for Similar Shapes*. Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'91), pp.208–217, 1991.

[JB91] Jagadish, H. V. and Bruckstein, A. M.: *On sequential shape descriptions*. Pattern Recocnition, 1991.

[Kei99] Keim, D.A.: *Efficient Geometry-based Similarity Search of 3D Spatial Databases*. Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA, pp. 419–430, 1999.

[KKV90] Kubicka, E. and Kubicki, G. and Vakalis, I.: *Using Graph Distance in Object Recognition*. Proc. ACM Computer Science Conference, pp. 43–48, 1990.

[KS03] Kriegel, H.-P. and Schönauer, S.: *Similarity Search in Structured Data*. Proc. 5th Int. Conf. DaWaK 2003, Prague, Czech Republic, LNCS 2992, pp. 676–693, 2004.

[KSF+98] Korn,K. and Sidiropoulos, N. and Faloutsos, C. and Siegel, E. and Protopapas, Z.: *Fast and effective retrieval of medical tumor shapes*. IEEE Transactions on Knowledge and Data Engineering, 10(6), pp. 889–904, 1998.

[KSS97] Kriegel, H.-P. and Schmidt, T. and Seidl, T.: *3D Similarity Search by Shape Approximation*. Proc. 5th Int. Symposium on Large Spatial Databases (SSD'97), Berlin, Germany, volume 1262 of Lecture Notes in Computer Science (LNCS), pp. 11–28, 1997.

[Kuh55] Kuhn, H.W.: *The Hungarian method for the assignment problem*. Naval Research Logistics Quarterly, 2:83-97, 1995.

[Lev66] Levenshtein, V.: *Binary codes capable of correcting deletions, insertions and reversals*. Soviet Physics-Doklady, 10, pp. 707–710, 1966.

[MG93] Mehrotra, R. and Gary, J.E.: *Feature-Based Retrieval of Similar Shapes*. Proc. 9th Int. Conf. on Data Engineering, Vienna, Austria, pp. 108–115, 1993.

[Mun57] Munkres, J.: *Algorithms for the assignment and transportation problems*. Journal of the SIAM, 6:32-38, 1957.

[Pet02] Petrakis, E.: *Design and evaluation of spatial similarity approaches for image retrieval*. Image and Vision Computing, 20(1), pp. 59–76, 2002.

[RB00] Ramon, J. and Bruynooghe, M.: *A polynomial time computable metric between point sets*. Acta Informatica, 37: 765-780, 2001.

[SF83] Sanfeliu, A. and Fu, K.-S.: *A distance measure between attributed relational graphs for pattern recognition*. IEEE Transactions on Systems, Man and Cybernetics, 13(3), pp. 353–362, 1983.

[SK98] Seidl, T. and Kriegel, H.-P.: *Optimal multi-step k-nearest neighbor search*. Proc. 17th ACM SIGMOD, Seattle, WA, pp. 154–165, 1998.

[SKSH89] Schneider, R. and Kriegel, H.-P. and Seeger, B. and Heep, S.: *Geometry-based Similarity Retrieval of Rotational Parts*. Proc. Int. Conf. on Data and Knowledge Systems for Manufacturing and Engineering, Gaithersburg, ML, pp. 150–160, 1989.

[WF74] Wagner, R. A. and Fisher, M. J.: *The String-to-String Correction Problem*. Journal of the ACM, 21(1), pp. 168–173, 1974.

[WFKM97] Wiskott, L. and Fellous, J.-M. and Krüger, N. and von der Malsburg, C.: *Face Recognition by Elastic Bunch Graph Matching*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7), pp. 775–779, 1997.

[ZSS92] Zhang, K. and Statman, R. and Shasha, D.: *On the editing distance between unordered labeled trees*. Information Processing Letters, 42, pp. 133–139, 1992.

Research Problems in Data Provenance

Wang-Chiew Tan*
University of California, Santa Cruz
Email: wctan@cs.ucsc.edu

Abstract

The problem of tracing the provenance (also known as lineage) of data is an ubiquitous problem that is frequently encountered in databases that are the result of many transformation steps. Scientific databases and data warehouses are some examples of such databases. However, contributions from the database research community towards this problem have been somewhat limited. In this paper, we motivate the problem of supporting data provenance in scientific database applications and provide some background on previous research. We also briefly describe the DBNotes prototype developed at UC Santa Cruz that can be used to "eagerly" trace the provenance and flow of relational data and describe some directions for further research.

1 Introduction

The word "provenance" means "origin" or "source" and is often used in association with a piece of art or literature. Indeed, Merriam-Webster [19] additionally defines provenance as "the history of ownership of a valued object or work of art or literature". Knowledge on the history of ownerships of a work of art is of great importance and is evidenced by the number of provenance projects that we can find today, including the Provenance Research Project at the Metropolitan Museum [23] and at the Harvard University Art Museums [21]. According to [21], provenance "can help to determine the authenticity of a work, to establish the historical importance of a work by suggesting other artists who might have seen and been influenced by it, and to determine the legitimacy of current ownership". The motivation for understanding the provenance of works of art is also also applicable to data we see on the Web. With the proliferation of data on the Web, questions such as "Where did this data come from?", "Who else is using this data?", and "Why is this piece of data here?" are becoming increasingly common.

The field of molecular biology has some 500 databases [14] but only a handful of these are source databases in the sense that they receive experimental data directly. In fact, most databases are *curated* from other databases, i.e., they consists of data integrated from several other databases, often with extensive manual input and cleansing performed by the maintainers of the database. In such a scenario, it is natural for a user to ask if a piece of data in the curated database is added by the maintainers or derived from other source database in order to help determine the quality of information that she is receiving. Despite the importance of understanding the provenance of data, there has been relatively few foundational techniques and tools that have been developed to help one trace the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Supported in part by an NSF CAREER Award IIS-0347065 and NSF grant IIS-0430994.

provenance of data in scientific databases. Data can be physically moved from one database to another as a result of executing a query, a program or a manual transformation or virtually moved from one database to another as a result of high-level descriptions of the relationships that are specified between data sources. Unless there is a concerted effort to explicitly maintain provenance as data is moved around, such information is usually lost in a database transformation process. Also, the situation is often complicated by the fact that a source database may be evolving in both its content and structure over time. A complete description of data provenance in this scenario would involve tracing a piece of data to its correct source version and locating that piece of data in other versions, if any. This means that every version of the source database that ever existed must somehow be retained and the relationships between different versions, which may differ in both structure and content, must either be derivable or documented.

It should be noted that this paper is devoted to the discussion of the provenance of data, as opposed to the provenance of a data product even though both types of provenance are equally important [3, 16]. The latter deals with how a data product is derived. For example, the provenance of the result from a scientific experiment may include the workflow steps, software and the version of software, software parameters, the operating system, and raw data that are used in the experiment to arrive at the result.

In the next section, we provide some background on existing work related to data provenance and in Section 3, we describe a few applications where techniques for tracing data provenance are required. In Section 4, we briefly describe the DBNotes prototype developed at UC Santa Cruz and in Section 5, we describe some directions for further research.

2 Background

Existing techniques for tracing data provenance can be roughly classified under two approaches: "lazy" or "eager". Techniques for archiving data can generally be classified under two approaches: sequence-of-delta or timestamping.

2.1 The Lazy vs. Eager Approach to Tracing Data Provenance

The lazy approach computes the provenance of data only when needed. Existing work based on this approach generally assumes that the transformation process between the source database and the target database is available and is given as a query Q. The query Q and the output data d, whose provenance is of interest, are analyzed when needed and a new query is generated based on the analysis. When the new query is evaluated against the source database, it computes the provenance of d which are combinations of source tuples that together with the query Q, derive d. Existing techniques that uses this approach include [5, 11, 12, 31]. The problem of computing data provenance in the relational model was first considered in [31]. Provenance is traced at the attribute level and can be combined to obtain the provenance of data at a higher granularity (e.g., a tuple). The function that computes the output is "inverted" but this inverted function is a weak inverse in general; in addition to all the right answers returned, it may also return some wrong answers. A separate verification phase is used to remove the wrong answers. In [12], provenance is computed by analyzing queries in the relational algebra framework. A new query, which computes the provenance when it is applied to the source databases, is automatically generated from the analysis. Techniques for computing data provenance under general data warehouse transformations were proposed in [11]. In [5], an algorithm that can compute the provenance of a piece of data in a hierarchical deterministic data model in the result of a query was given. In addition, two kinds of provenance, why and where-provenance, were distinguished and characterized under this model. Intuitively, the *why-provenance* of a piece of data d in the result consists of sets of minimal pieces of source data such that each set, together with the query, is sufficient to reconstruct d in the output. On the other hand, the *where-provenance* of a piece of data d in the result is a more refined notion that tells us the exact pieces of source data where d is copied from.

Another approach to compute data provenance is to do so eagerly by carrying the provenance of data along as data is transformed [1, 2, 6, 17, 26, 29]. This approach has been given several names such as metadata support, source tagging, the attribution approach, or annotations. An advantage of using an eager method to compute data provenance is that the source databases need not be probed since the provenance can be fully determined by looking at the annotations associated with a piece of output data. A disadvantage is that the eager approach incurs additional overhead to compute the output and store annotations in the output. In Section 4, we shall describe the DBNotes prototype developed at UC Santa Cruz which allows one to eagerly propagate provenance along as data is transformed.

2.2 The Sequence-of-Delta vs. Timestamping Approach to Archiving

Many approaches for archiving data use the sequence-of-delta approach which stores a reference version (i.e., the first version or last version) and a sequence of forward or backward deltas between successive versions [9, 13, 18, 27]. For example, given a sequence of database versions V_1, V_2, ..., V_k, the sequence-of-delta approach may store V_1, together with Δ_{12} and Δ_{23}, ..., $\Delta_{k-1,k}$ where Δ_{ij} is the difference from version i to version j. Version V_1 together with the sequence of deltas Δ_{12}, ..., $\Delta_{i-1,i}$, is sufficient to reconstruct version i (and versions 1 through $i - 1$ for that matter). Ideally, each delta is the smallest possible difference between two versions. A delta is typically computed with differencing techniques whose goal is to minimize the size of the delta or compute a reasonably small delta efficiently. A significant number of efficient algorithms for computing a delta, depending on the type of the file and application, have been proposed [7, 8, 10, 20, 22, 25, 32, 33].

In the timestamping approach, a "big" repository stores every version and timestamps are used to mark the existence of data elements at various times. For example, the source code control system (SCCS) [24] uses this technique. Given two versions to be archived, the difference (or correspondence of lines of text) between two versions is first computed. The difference is then used to timestamp lines of text in the repository to denote the addition or deletion of various lines of text at different times. The sequence-of-delta and timestamping approaches differ only in their change representation: the former technique describes the difference between two versions around time (through a delta) while the latter technique describes the difference around data (through timestamps).

These approaches which are based on diff often fail to capture the underlying semantics of data. As an example, if an object has changed some of its attribute values over time, a diff algorithm might erroneously report the change as that object exchanging its identity with another object [4]. This anomaly does not happen, however, if we only retrieve entire versions from the repository. It matters only when we are interested in retrieving changes pertaining to an object in the database. In [4], an archiving technique that avoids this anomaly by exploiting hierarchical key constraints has been developed.

3 Application Scenarios

In what follows, we provide some example scenarios where techniques for tracing the provenance of data and archiving are applicable.

Gauging the Trustworthiness of Data In an environment where data is repeatedly copied or edited on its journey from a source database to a new database, users are often interested to know the provenance of data in order to gauge the trustworthiness of data. Many scientific databases or biological databases, for example, fall into this scenario. These databases are often curated. For a user of such databases, it is crucial to understand whether the information she encounters in the database is added by the maintainers of that database, or derived from other databases for she may trust certain sources of information more than the others. This can also help

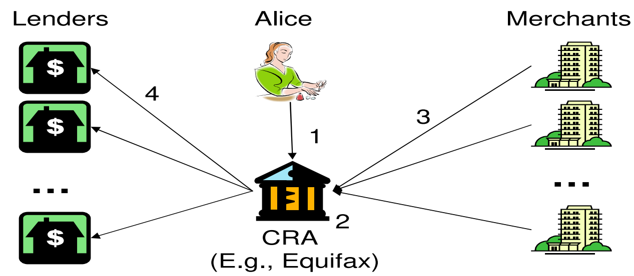


Figure 1: A complete example with the credit bureau.

determine the sources of error, if any, and gives a better understanding on the overall quality of the curated database.

Sharing Knowledge via Annotations There is usually a group of users associated with each database. It is common for these users to find errors, inconsistencies in the data, or discover additional information about the data and want to share their knowledge and experiences with other users. Since the database schema is often fixed and proprietary, one cannot simply modify the database to insert additional information that a user may have. A common practice nowadays is for the user to post her findings in a mailing list or newsgroup relevant for the community and hopefully, the changes will be reflected in the subsequent version of the database. However, unless every user makes a habit of reading and recording such announcements, it is likely that some valuable information will be lost in the long run.

Ideally, to share knowledge about a piece of data among users of the same community, one should be able to overlay the annotations on that piece of data and these annotations would automatically be reflected to other users who are looking at the same piece of data. The structure of annotations is not constrained by the schema of the database, if it exists. With a collaborative tool that allows annotations to be added and reviewed by many independent parties, it is likely that the quality of data in the database will be higher in the long run.

Verifying Data Recently, it is common in scientific literature to use references to Web pages to substantiate the claims made in the literature. The credibility of these claims therefore relies on the contents of the Web pages that have been cited. It happens frequently, however, that the contents of these Web pages change over time or become unavailable and hence, the claims are no longer verifiable. For scientific databases, research findings are often based on data of a particular version and it is therefore crucial to keep every version of the database that ever existed. The same reason applies to many other databases, such as financial databases or stock inventory databases, where a financial report or market-trend analysis may have been based on a particular version of the database and we would like these reports and analysis to remain verifiable.

3.1 Putting it all together: A Complete System for Tracing Data Provenance

We next describe a credit bureau example that would require one to apply techniques for tracing data provenance, annotation propagation and archiving in a single business process. See Figure 1.

1. A credit bureau[1] e.g., Equifax, receives a letter from Alice claiming an inaccuracy in her credit history report, dated two months ago.

2. Equifax retrieves Alice's record from their archive in order to verify her claim. This step involves tracing for Alice's two month old record in the archive and verifying her claim against the record in the database.

[1] Credit bureaus, also known as credit reporting agencies (CRAs), collect information from our creditors (e.g., merchants, landlords, etc.) about our credit habits and then sell this information to businesses (e.g., lenders) so they can evaluate our application for credit.

At the same time, Equifax also traces the record in the archive forward in time in order to verify that Alice's record has not been updated with the correct information since.

3. After having verified that Alice's record is indeed as claimed and has not been corrected, Equifax then checks for the provenance of the disputed entry in which it attempts to identify the merchant responsible for reporting the disputed entry. Equifax found that the merchant responsible for reporting the disputed entry has a known history of inaccurate reporting.

4. A formal investigation is then initiated by Equifax to investigate the dispute. Meanwhile, Equifax annotates the disputed entry in Alice's record with Alice's written statement of dispute so that potential lenders who are reviewing Alice's credit history are aware of the dispute.

The above example shows that in order for Equifax to smoothly handle the dispute from Alice, it requires an archival database system that has the capability of reporting the history of a record (in this case, a trace on Alice's record was made as described in Step 2), tracing the provenance of a disputed entry (Step 3) and propagating annotations of a record through views to the lenders (Step 4).

4 The DBNotes Project at UC Santa Cruz

DBNotes is an annotation management system developed at UC Santa Cruz that currently works on top of a relational database system. In DBNotes, every value can be associated with zero or more annotations. A *value* refers to an attribute value of a tuple in some relation. Annotations are automatically propagated along with the values as they are being transformed through a query. Currently, DBNotes supports the propagation of annotations through an extension of a fragment of SQL, called pSQL. In its default behavior, DBNotes propagate annotations based on where data is *copied* from. As a consequence, if every column of every tuple in a database is annotated with its address, the provenance of data is propagated along as data is transformed. An example is shown below. Suppose we have the following tables SWISS-PROT and GENBANK.

SWISS-PROT

ID	Desc
q229 $\{a_1\}$	CC $\{a_2\}$
q939 $\{a_3\}$	ED $\{a_4\}$

GENBANK

ID	Desc
g231 $\{a_5\}$	AB $\{a_6\}$
g756 $\{a_7\}$	CC $\{a_8\}$

Q_1:

```
select distinct Desc
from SWISSPROT
propagate default
union
select distinct Desc
from GENBANK
propagate default
```

Result of Q_1:

Desc
AB $\{a_6\}$
CC $\{a_2, a_8\}$
ED $\{a_4\}$

Each value has a distinct address and is denoted by the annotation "a_i", $1 \leq i \leq 8$. The query Q_1 takes the union of the gene descriptions in SWISS-PROT and GENBANK. The `propagate default` clause specifies that annotations should be propagated in the default way, i.e., according to where data is copied from. Since the gene description `CC` is common to both SWISS-PROT and GENBANK, the corresponding addresses a_2 and a_8 of `CC` in SWISS-PROT and GENBANK respectively are merged together in the result of Q_1.

In general, an annotation is not restricted to be an address, it can also be a comment about a value such as its correctness and quality. As these comments are carried along query transformations, they give a sense of the overall quality of the resulting database that is generated by the query. DBNotes also allow the users to specify exactly how the annotations should propagate. This is done using the custom propagation scheme of pSQL. For example, the query Q_2 propagates the annotations associated with each ID value and Desc value of a tuple to the output Desc value according to the custom propagation scheme which is specified as "`propagate ID TO Desc, Desc TO Desc`".

Q_2:
select distinct Desc
from SWISSPROT
propagate ID TO Desc,
Desc to Desc

Result of Q_2:

Desc
CC $\{a_1, a_2\}$
ED $\{a_3, a_4\}$

Q_3:
select distinct s.Desc
from SWISSPROT s,
GENBANK g
where s.Desc = g.Desc
propagate default-all

Q_4:
select distinct g.Desc
from SWISSPROT s,
GENBANK g
where s.Desc = g.Desc
propagate default-all

Result of Q_3 or Q_4:

Desc
CC $\{a_2, a_8\}$

There is also a third propagation scheme supported by DBNotes, called the default-all propagation scheme. In this propagation scheme, annotations are propagated according to where data is copied from according to *all* equivalent queries. The reason for introducing the default-all scheme is because two equivalent SQL queries may not always generate the same annotations in the results under the default propagation scheme. On the other hand, the default-all propagation scheme gives the user an invariant semantics; two equivalent SQL queries under the default-all propagation scheme will always generate the same annotations in the results. The queries Q_3 and Q_4 above are equivalent but they generate different annotations in results under the default scheme (Q_3 returns the annotation a_2 while Q_4 returns the annotation a_8). However, they generate the same annotations in the results under the default-all propagation scheme. The result is shown above on the right. We refer the interested reader to [2] for more details about DBNotes. We note that recently, there is also interest in building a database system for managing and querying the accuracy and lineage of data [30].

5 Directions for Further Research

Motivated by the need to trace the provenance of data in scientific databases, we have identified some challenges below where we believe that research advances in these areas are important steps towards the overall goal of tracing the provenance of data in scientific databases.

Tracing Provenance through General Transformations Earlier works on tracing provenance have been largely database language and model specific. It happens often, however, that a database is created as a result of executing a program script or interpreting high-level abstractions given between two data sources or manual data entry. A complete system for tracing provenance should be able to handle these general transformations as well. At a minimum level, one should be able to treat a transformation step as a black-box and explain that a piece of data is generated by that transformation step using certain parameters. On the other end of the spectrum, one could also go at length to explain how a transformation step derived a piece of data. For data that has been manually entered, a complete description of provenance should involve the author, date and documentation on why that entry was made.

- **Computing provenance lazily** In this direction of research, a detailed analysis of a transformation step to compute the provenance of data is made only when the provenance is sought for. It would be interesting to see how earlier research ideas for tracing provenance in database model and language specific settings and research in *program slicing* [15] can be applied to the problem of computing the provenance of data in the result of a procedure or program execution.

- **Computing provenance eagerly** In this direction of research, the provenance of data is computed as data is being transformed. In this way, the provenance of any data is immediately available in the result when sought for. As discussed in Section 2.1, initial research along this direction has been proposed but further work on formalizing the semantics and properties of such systems is required.

- **Querying Provenance** Complementary to the problem of computing provenance is the problem of querying provenance. Examples of such queries include “Find all records generated by Alice at the Zurich Genetic Laboratory” or “Who else is looking at this record?”. These queries are not always straightforward to answer since the provenance of each record entry may not have been stored in the derived database. We may also require extensions to existing query languages to issue such queries.

Semantic Archiving and Querying Many databases are often well-structured and organized and come with semantic constraints (e.g., schema, key or foreign key constraints). Most existing techniques for archiving or versioning data, however, fail to exploit these properties as described in Section 2.2. Classical diff techniques rely on computing the minimum edit distance between two versions and storing the smallest "delta" that would give rise to the minimum edit distance. The end result is an archive that is the most compact and gives good performance for retrieving historical versions. Unfortunately, the archive is not always amenable to reasoning about the changes that has occurred to each semantic object.

An ideal archiver should not only be able to give a compact representation of historical versions of data but also exploit the semantic constraints so that the continuity of each semantic object is either preserved in the archive or easily reconstructed. Recent work [4] is one step towards semantic archiving and needs to be extended to handle more general settings.

Complementary to semantic archiving is the ability to pose historical queries to the archive. Examples of such queries include "retrieve Alice's record in 1998", "report the changes that has occurred to Alice since 2000", or "retrieve the date when Alice moved from Seattle to Santa Cruz". There are some connections to research in temporal databases [28], and needs to be explored.

6 Concluding Remarks

We have motivated the data provenance problem for scientific databases and general database applications. We have also described some previous research work on data provenance from the database research community and briefly described the DBNotes prototype developed at UC Santa Cruz. Although there has been some existing work on data provenance, we believe that there are still many important problems that are not satisfactorily solved and deserve attention for further research.

References

[1] P. Bernstein and T. Bergstraesser. Meta-Data Support for Data Transformations Using Microsoft Repository. *IEEE Data Engineering Bulletin*, 22(1):9–14, 1999.

[2] D. Bhagwat, L. Chiticariu, W. Tan, and G. Vijayvargiya. An Annotation Management System for Relational Databases. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 900–911, Toronto, Canada, 2004.

[3] P. Buneman and I. Foster. Workshop on Data Derivation and Provenance. http://www-fp.mcs.anl.gov/˜foster/provenance/.

[4] P. Buneman, S. Khanna, K. Tajima, and W. Tan. Archiving Scientific Data. *ACM Transactions on Database Systems (TODS)(SIGMOD/PODS Special Issue)*, 29(1):2–42, 2004.

[5] P. Buneman, S. Khanna, and W. Tan. Why and Where: A Characterization of Data Provenance. In *Proceedings of the International Conference on Database Theory (ICDT)*, pages 316–330, London, United Kingdom, 2001.

[6] P. Buneman, S. Khanna, and W. Tan. On Propagation of Deletions and Annotations Through Views. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 150–158, Wisconsin, Madison, 2002.

[7] S. S. Chawathe and H. Garcia-Molina. Meaningful Change Detection in Structured Data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 26–37, Tucson, Arizona, 1997.

[8] S. S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom. Change Detection in Hierarchically Structured Information. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 493–504, Montreal, Canada, 1996.

[9] S. Chien, V. Tsotras, and C. Zaniolo. Efficient Management of Multiversion Documents by Object Referencing. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 291–300, Roma, Italy, 2001.

[10] G. Cobena, S. Abiteboul, and A. Marian. Detecting Changes in XML Documents. In *Proceedings of the International Conference on Data Engineering (ICDE)*, Heidelberg, Germany, 2001.

[11] Y. Cui and J. Widom. Lineage Tracing for General Data Warehouse Tranformations. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 471–480, Roma, Italy, 2001.

[12] Y. Cui, J. Widom, and J. Wiener. Tracing the Lineage of View Data in a Warehousing Environment. *ACM Transactions on Database Systems (TODS)*, 25(2):179–227, 2000.

[13] Concurrent Versions System: The open standard for version control. http://www.cvshome.org.

[14] DBCAT, The Public Catalog of Databases. http://www.infobiogen.fr/services/dbcat/, cited 5 June 2000.

[15] S. Horwitz, T. Reps, and D. Binkley. Interprocedural slicing using dependence graphs. *SIGPLAN Notices*, 39(4):229–243, 2004.

[16] R. Hull. E-services: a look behind the curtain. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 1–14, San Diego, California, 2003.

[17] T. Lee, S. Bressan, and S. Madnick. Source Attribution for Querying Against Semi-structured Documents. In *Workshop on Web Information and Data Management (WIDM)*, Washington, DC, 1998.

[18] A. Marian, S. Abiteboul, G. Cobena, and L. Mignet. Change-Centric Management of Versions in an XML Warehouse. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 581–590, Roma, Italy, 2001.

[19] Merriam-Webster. Merriam-Webster Online - The Language Center.
http://www.m-w.com/home.htm.

[20] W. Miller and E. Myers. A file comparison program. *Software-Practice and Experience*, 15(11):1025–1040, 1985.

[21] H. U. A. Museums. Provenance Research.
http://www.artmuseums.harvard.edu/research/provenance/index.html.

[22] E. Myers. An $O(ND)$ difference algorithm and its variations. *Algorithmica*, 1(2):251–266, 1986.

[23] T. M. M. of Art. Provenance Research Project.
http://www.metmuseum.org/collections/department.asp?dep=22&full=1.

[24] M. Rochkind. The Source Code Control System. *IEEE Transactions on Software Engineering*, 1(4):364–370, 1975.

[25] K. C. Tai. The tree-to-tree correction problem. *Journal of the Association for Computing Machinery (JACM)*, 26:422–433, 1979.

[26] W. Tan. Containment of relational queries with annotation propagation. In *Proceedings of the International Workshop on Database and Programming Languages (DBPL)*, Potsdam, Germany, 2003.

[27] W. Tichy. RCS - A system for Version Control. *Software – Practice and Experience*, 15(7):637–654, 1985.

[28] K. Torp, C. S. Jensen, and R. T. Snodgrass. Effective Timestamping in Databases. *The VLDB Journal*, 8(3-4):267–288, 2000.

[29] Y. R. Wang and S. E. Madnick. A Polygen Model for Heterogeneous Database Systems: The Source Tagging Perspective. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 519–538, Brisbane, Queensland, Australia, 1990.

[30] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. Technical report, Stanford University, 2004. http://dbpubs.stanford.edu/pub/2004-40.

[31] A. Woodruff and M. Stonebraker. Supporting fine-grained data lineage in a database visualization environment. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 91–102, Birmingham, United Kingdom, 1997.

[32] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18(6):1245–1262, 1989.

[33] K. Zhang and D. Shasha. Fast algorithms for unit cost editing distance between trees. *Journal of Algorithms*, 11(6):581–621, 1990.

Towards an Integrated Software System for Biological Pathways

Z. M. Ozsoyoglu[1,2], J. Nadeau[1,3], G. Ozsoyoglu[1,2], M. Tasan[1,2]

[1]Center for Computational Genomics
[2]Department of Electrical Engineering and Computer Science
[3]Department of Genetics

Case Western Reserve University, Cleveland, Ohio
(meral, joseph.nadeau, tekin, murat.tasan)@case.edu

Abstract

In this paper, we describe the Pathways Database System, being developed at Case Western Reserve University. Pathways Database System is an integrated software system for storing, managing, analyzing, and querying biological pathways at different levels of genetic, molecular, biochemical and organismal detail. We summarize the data model and the architecture of the Pathways Database System, and then briefly discuss two recently developed Pathways Database System tools, namely, the Pathway Explorer (a tree-structured query interface), and the Pathway Editor (to create and update pathways). We present a list of database research problems associated with pathways, and conclude with a summary of future work on the Pathways Database System.

1 Introduction

Biological pathways are used for presentation, modelling genomic data, studying genome sequences, and in functional studies of attributing functions to genes. Pathways are the sequential and cumulative action of genetically distinct, but functionally related molecules. Each reaction in a pathway is a biochemical step from specific substrates (input molecules) to products (output molecules) that are chemically modified substrates. Each step may also use combinations of molecules as cofactors, activators, inhibitors, and regulators, and usually involves at least one genetically unique gene product that catalyzes the reaction step.

The three general classes of biological pathways are metabolic and biochemical pathways, transcription and protein synthesis pathways, and signaling pathways. Metabolic pathways are the most known pathways. They involve the chemical reactions that provide basic biological functions such as energy metabolism, fatty acid synthesis, glycolysis, citric acid cycle, and many others. Transcription and protein synthesis pathways refer to synthesis of proteins from DNA. Signaling pathways are responsible for coordinating metabolic processes and other molecular and cellular activities, and transcription and protein synthesis processes.

Pathways, in general, illustrate the functional relations between molecules. Functional annotations include, for example, the identity of the substrate(s), product(s), cofactors, activators, inhibitors, enzymes or

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

other processing molecules, RNA and protein expression patterns, reaction kinetics, and associated phenotypic variation and diseases. Ultimately, many other kinds of information (or knowledge) can be incorporated. Such information forms a rich research resource that integrates genomic and biological information which can be managed, analyzed, queried and displayed in dynamic ways at various levels of biological and genetic detail to provide insight into diverse biological processes in health and disease.

Pathways Database System[K+a03] is an integrated software system for storing, managing, analyzing, and querying biological pathways at different levels of genetic, molecular, biochemical and organismal detail. At the computational level, the system allows users to visualize pathways in multiple abstraction levels, and to pose predetermined as well as ad hoc queries using a graphical user interface. Pathways are represented as graphs, and implemented on top of a relational database. Pathways Database System has multiple levels, with multiple tools at each level. In this paper, we summarize Pathways Database System's data model and architecture in general. We then present in more detail Pathway Explorer (a Tree-structured Query Interface. We conclude with what we view as the database research problems about pathways systems, and our future work.

2 Data Model

The design of the underlying database in the Pathways Database System is the primary influence on how data is to be acquired, presented, edited, and entered. While different tools can provide slightly different approaches, the logical steps for handling the data must be the same, and ultimately this is reflected in an overall similarity in all tools that access and work with the database. In what follows, we describe, in very basic ways, the underlying biological terminology as well as the corresponding entities and relationships captured in our data model. For more details on the data model, see [T04].

Molecular Entities. All biochemical pathways can be simply described as molecular entity interactions throughout the organism. Basically, molecules that are floating around in (and out of) the cell sometimes get together, resulting often in changes to the molecules. These changed molecules then go off and interact with other molecules, continuing the procedure, and thus giving birth to cascading pathways. So, at the very foundation of the notion of a pathway is the single molecule, which is referred to here as a molecular entity. Lumping all molecular entities together though into a single group is not a good modeling practice as different classes of entities (e.g., RNAs, amino acids) possess unique attributes. Therefore, the molecular entities are broken up into different classes, e.g., genes, RNAs, proteins, amino acids, and basic molecules.

Processes. The process refers to an interaction between molecular entities. A process is also sometimes seen as a single step (or reaction) in a metabolic pathway. Most of the chemical reactions that are included in metabolic pathways are catalyzed by some particular enzyme. There are also cases where no catalyzing enzyme (or catalyzing RNA) is present, and the reactants engage in a process entirely on their own. In either sort of process, there are a series of roles that different molecular entities can play. For each process in the Pathways Database, all of the known reactants are stored (with associated quantity attributes, to allow for correct stoichiometry), as well as some other process-specific attributes (such as whether or not the process is reversible, cellular location, etc.).

Each molecular entity involved in a process falls into some category or role-players for that process. The major categories of roles are those entities that are consumed (or destroyed), those that are created (or produced), and those that are not modified. Catalyzing enzyme allows molecules that would otherwise not interact to be brought into contact with each other in such a way that a reaction occurs. Once the process is complete, the catalyzing enzyme releases any reactants and moves on, completely unmodified (hence the term catalyzing enzyme). The next two roles are the substrates and the products, which are the primary reactants in the process. If a molecular entity is described as a substrate in a process, it will be consumed during the course of the reaction, and will not be present (at least in the same quantity) after the process has completed. Similarly, a product is not present at the start of the process, but will be at the end. It is through the substrates and the products that most pathways are defined. Cofactors are secondary substrates (cofactor-in) and secondary products (cofactor-out). Cofactors are also consumed or produced during the course of a process. There are also other molecular entities that can be involved in the reaction that are not modified besides the catalyzing protein. They are often called regulators when it is not entirely clear how they affect the process. If the reaction is dependent on a regulator, or if the regulator speeds up the rate of the reaction, they are usually called activators. On the other hand, inhibitors are regulators that play a negative role in the completion (or initiation) of the process.

Some processes are reversible, and are thus governed by concentration equilibria usually. Since we must store the roles of the process in at least one manner, for those reactions that are reversible it is clear that the choice is somewhat arbitrary. When a process is reversed, the entire set of substrates can become the set of products, and vice versa. The same dichotomy exists for cofactors (in becomes out, out becomes in), as they are functionally often the same as products and substrates. For regulators, it is not always clear if an activator will become an inhibitor and vice versa, and indeed since it is not fully understood biochemically, it can only be modeled if the understanding of its behavior is available.

Next we describe the data modeling of two pathways classes, namely, metabolic and signaling pathways.

Metabolic Pathways. The metabolic pathway is where we can see the processes in action; a pathway is simply a collection of processes. Typically, interrelated processes to describe a particular aspect of life function are grouped into a pathway, and named accordingly, e.g., fatty acid synthesis pathway.

Logically, a metabolic pathway becomes a graph of processes, with edges between them if there exists a substrate/product relationship between the processes (i.e. a product of one process is used as a substrate of another). A disconnected graph can be a flag for a possible error, e.g. maybe the curator should consider making each component a separate pathway itself. From the database perspective, a pathway is stored very easily with a few attributes (name, type, references, etc.) and a collection of the processes it consists of.

Signaling Pathways. In all multicellular organisms, an elaborate intercellular communication network coordinates the growth, differentiation, and metabolism of cells in diverse tissues and organs. Cells communicate in terms of extra *cellular signaling molecules*, which are synthesized and released by *signaling cells* and produce a specific response only in *target cells* that have *receptors* or *transporters* for the signaling molecules. A large variety of molecules, including proteins, peptides, etc. are used in cell-to-cell communication, called *signal transduction* (or *signaling*) *pathway*, that is, the process of converting extra cellular signals into cellular responses. Signaling pathways regulate many different aspects of cellular metabolism, function, and development. Specific signals promote changes in gene expression, cell morphology, and cell movements.

Communication by extra cellular signaling usually involves five parts (captured in our data model): synthesis (release of a signaling molecule by signaling cell), transport (of signal to target cell), signal detection (by a specific receptor protein), a change in cellular metabolism/function, and removal of signal to terminate further cellular response. Based on the operation distance of signaling molecules, there are three types of signaling pathways: *endocrine signaling pathways* (hormones carried over blood into target cells), *paracrine signaling pathways* (signaling molecules having an effect on cells in close proximity), and *autocrine signaling pathways* (cells responding to molecules released by themselves).

Signaling pathways may be viewed as consisting of multiple *signaling steps* where one or more *signaling molecules* signal one or more *signaled molecules*. It is possible that a signaling step may involve the substrate and the product, or enzyme of a metabolic pathway process. Molecule locations are captured and involve tissue (loosely defined as a collection of cells performing a specified function) and organelle information (i.e., the membrane, cytoplasm, nucleus membrane, or nucleus). Locations of signaling/signaled molecules may be different. Thus a signaling step can be viewed as a (usually single) directed edge from signaling molecules to signaled molecules, enhanced by location information, where location consists of organelle and tissue information.

A large number of information about signaling molecules can be captured in a data model. While our data model continually evolves, presently we capture location, role (e.g., enzyme, hormone, receptor, transcription factor, etc.), molecule families (super/subfamily), and component/subcomponent information, among others.

In comparison with metabolic pathways, there is much less agreement as to what signaling steps together constitute a specific signaling pathway with a certain biological function. Our present solution to this problem is to name and represent all proposed variations in the database.

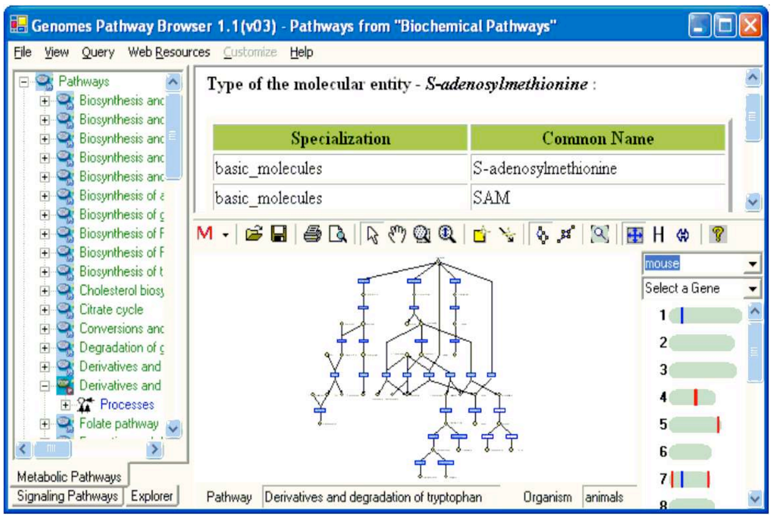


Figure 1: Snapshot of the Pathway System main GUI. The leftmost panel is the Pathway Browser (metabolic pathway or signaling pathway). The upper-right pane displays query results. The lower-right pane is the Pathway Viewer within the main GUI (it can also be a separate window).

3 Architecture

Pathway Database System is developed using the .Net platform (a Java version is also being developed [R03, PSRJ03]). The system architecture has three layers. At the top layer, we have a set of GUI tools that are continually improving and evolving, and increasing in number. At the middle layer, we have two service subsystems, Querying Services and Data Extraction Services. At the lowest level, the Pathways Database, currently a relational database, resides. Next we discuss only the top-level tools. For a discussion of the other tools, please see [K+a03] and the technical reports of Pathway System Release #2 [PSR04].

The user interface tools at the top level are all presented through a single interface, with the only exception that users can visualize multiple pathways in multiple separate windows. However, all the tools are developed independently as plug-ins to the main interface, and communicate with each other at the user command level asynchronously. At the present time, we have developed four tools, which are described next.

Pathway Browser provides a tree-structured view of the metabolic pathways, and signaling pathways available in the database. Metabolic pathways, processes and molecular entities can be browsed in different orders (Figure 1, leftmost pane), and a set of useful queries [X04] can be specified about pathways, processes and molecular entities, from predefined simple queries to parameterized path selection and neighborhood queries. Similarly, for signaling pathways, signaling steps, and signaling and signaled molecules and their components can be browsed, and a set of useful queries are presently being developed. For more details, see [K03].

Pathway Viewer visualizes a single pathway (Figure 1, lower middle pane) or multiple pathways in an organism-specific or organism-independent manner, provides a large number of collapsing-zooming, hiding-expanding, etc. functionalities (Figure 1, top of middle pane), allows XML versions of pathways in the database to be exported, saved, and loaded back to the viewer and re-visualized. It also provides a seamless transition from metabolic pathways to signaling pathways, and from signaling steps that belong to metabolic pathway processes into the relevant pathways. Pathway viewer for signaling pathways is currently under construction. For more details, see [M04].

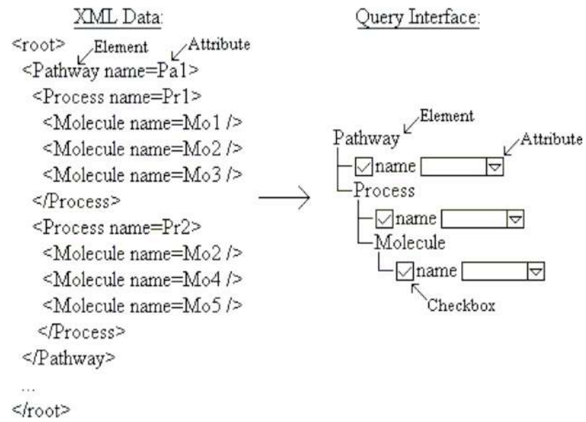


Figure 2: From Schema Elements to the Tree-Structured Interface

Pathway Explorer provides a generic user interface for posing a large class of queries about pathways using tree-structured views; see section 4 below.

Pathway Editor allows view-based database updates; that is, users can insert/update/edit pathways via the high-level user views of the pathway viewer. The pathway editor, while enforcing the integrity of the existing database (sometimes interactively with the user), maps them as updates to the underlying relational database. See section 5 below.

4 Pathways Explorer: A Tree Structured Query Interface

As it is clear from the discussion on the data model, entities involved in the pathways database can be better viewed by generalization/specialization hierarchies, e.g., classes of pathways, super pathways, functional classifications in arbitrary granularities, etc.. In addition, pathway components, i.e., processes in a pathway, and molecular entities involved in each process can naturally be viewed in the form of a tree-structured (hierarchical) view. Motivated by this observation, Pathways Explorer is a query interface which allows users to query the database using tree-structured views. As an example, consider the tree-structured (hierarchical) schema Pathways-Process-Molecular entity, where a pathway consists of several processes, and each process is associated with several molecular entities. This may also be visualized as a logical schema of a nested relation (Pathways (Process (Molecular_Entity)*)*), or an XML view. Thus, Pathways Explorer can also be viewed as a query interface for querying tree-structured views for a relational database, as shown in Figure 2.

Given the schema of an XML view, or a nested relation view of the underlying relational database, query interface (template) structure follows that of the view schema. As shown in Figure 2, the schema declares that pathway elements contain process elements that contain molecule elements, so the tree in the user-interface would contain only these nodes, along with the node attributes as children. Each attribute's possible values may then be seen individually by opening its drop-down box. Choosing a value for an attribute effectively creates a query constraint declaring that only nodes with the chosen attribute value are of interest. As the user chooses more constraints, the number of choices in each drop-down box decreases because the source XML view is being dynamically trimmed in the background as the constraints narrow its content (in reality, there is no actual trimming as this would be inefficient; but this paradigm makes the behavior easy to grasp).

In general, the tree in the interface need not have a one-to-one correspondence to that of the schema. The user may construct the tree (and effectively the output as well) using any of the elements in any arrangement. In the above example, the structure of the interface may instead be a molecule associated with a process which is in turn associated with a pathway (i.e., molecule is the root; pathway is the leaf). It could also be a process in a

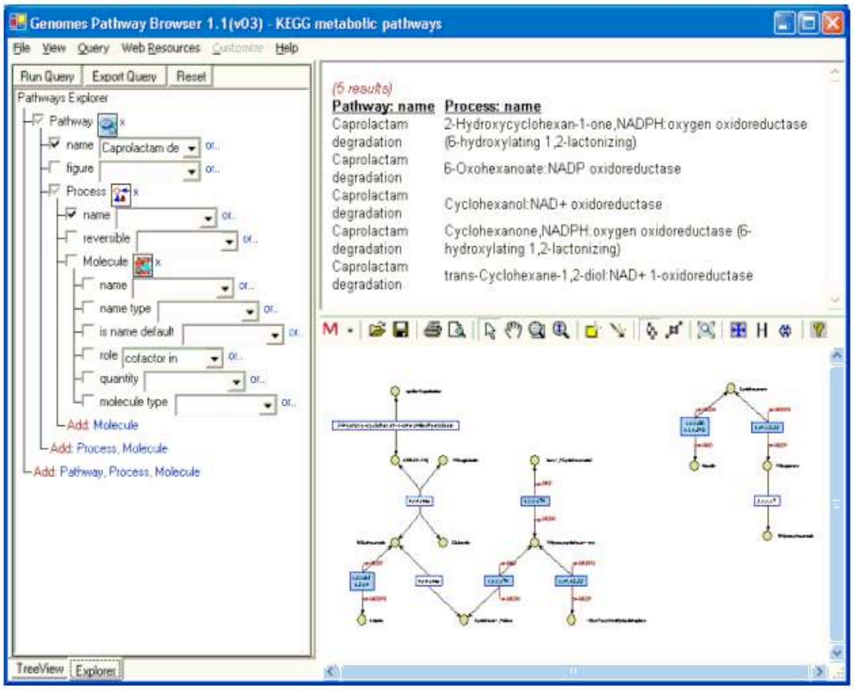


Figure 3: From Schema Elements to the Tree-Structured Interface

pathway and containing a molecule (i.e., process is the root; and pathway and molecule are its two children). As long as there is an ancestor-descendent relationship between two elements in the schema, these elements may be in a parent-child or child-parent relationship in the tree.

The Pathway Explorer interface has been implemented [NO04]. Query capabilities include selection, projection, equijoins, and self-joins. Figure 3 illustrates a query using Pathway Explorer, where the left pane shows the tree-structured query template, top pane displays the query results, and the bottom right pane shows graphs of the user-requested biological pathways or processes. For more details, see [NO04].

5 Pathway Editor Tool

Pathway Editor allows users to create or update pathways in the pathways database. The basic issues with and the properties of the Pathway Editor are listed below.

- *Transaction-oriented operations*. Operations such as creating or editing a pathway or a process are multi-level, multi-step operations with each step at each level involving database updates. It is possible that, at an intermediate step, the overall operation needs to be cancelled and all previous database updates are rolled back. Thus, all pathway editor updates are implemented as atomic transactions.

- *Interactive Communication*. The pathways database contains a myriad of biologically-inspired integrity constraints, such as the cofactor-in ATP in a process is accompanied by the cofactor-out ADP, or pathways are usually connected, or the requested update for a certain organism is not permissible, etc. As users perform a step in an update, the associated integrity constraint can be satisfied in multiple ways. This necessitates an interactive communication between the Pathway Editor and the users, with the Pathway Editor deciding on which route to take and the users ultimately controlling the update route and the enforcement of the integrity constraints.

- *Extensibility and Flexibility*. The Pathway System is continually evolving with new pathway types and functionality, and new extensions to the underlying pathway data model. To allow for these extensions, the Pathway Editor is designed to be extensible and flexible.
- *Familiar User Interface*. The Pathway Editor has a user interface similar to the other tools in the system. consistent user interface reduces the study curve for users.
- *Minimum-effort pathway editing*. In most biological databases, data gathering and data editing are among the most difficult tasks, time-consuming and error-prone. Pathway Editor is continually improved to ease the user workload by providing pop-up and drop-down menus of the existing objects in the database, prompting users with reminder-type questions and steps, and implementing a step-by-step enforcement of data integrity-related issues.
- *Complexity Hiding*. As much as possible, Pathway Editor is designed to hide from users the complexity of the operations it performs associated with pathway updates.

6 Other Pathway Systems

For metabolic pathways, there are a large number of systems with varying capabilities, which include PathDB system for pathways (http://www.ncgr.org/pathdb/architecture.html), Kyoto Encyclopedia of Genes and Genomes (www.genome.ad.jp/kegg),Metabolic Pathways of Biochemistry (www.gwu.edu/ mpb), Encyclopedia of E. Coli Genes and metabolism (http://ecocyc.pangeasystems.com/ecocyc/ecocyc.html), Biocarta (www.biocarta.com) with statically available metabolic and signaling pathways, Enzymes and Metabolic Pathways (emp.mcs.anl.gov), AceDB (www.acedb.org), Soybean metabolic pathways (http://cgsc.biology.yale.edu/metab.html), Nicholson minimaps (www.tcd.ie/Biochemistry/IUBMB-Nicholson), ExPASy Biochemical pathways (www.expasy.ch/cgi-bin/search-biochem-index), and others. These systems range from simply displaying picture images of metabolic pathways to database-backed advanced systems. Among the systems listed above, in terms of capabilities and tools, PathDB is closest to our Pathways Database System with the QueryTool (similar to our Pathway Browser), the Pathway Viewer, and the Discovery Tool (that provides data analysis approaches). PathDB however only deals with metabolic pathways.

On the signaling pathways side, the existing systems include CSNDB (Cell Signaling Networks database at http://geo.nihs.go.jp/csndb), TRANSPATH (http://transpath.gbf.de), Roche Apoptosis Pathway (www.grt.kyushu-u.ac.jp/spad), Database of Quantitative Cellular Signaling (DQCS, at http://doqcs.ncbs.res.in), Signaling Pathway database (SPAD, at www.grt.kyushu-u.ac.jp/spad/index.html). All of these systems model a large number of biological and pathways information; however, they are specifically for signaling pathways.

7 Database Research Problems

We have found early on that building biological data content and data curation (which is better done by a life scientist who is familiar with computer science), data provenance and data integrity enforcement are among the most difficult and important issues, if not the most important one, with pathway systems. Pathways data in biological resources and on the web are incomplete, and curation requires expert knowledge and data extraction. Moreover, as with any biological information, all pathways-related rules have various exceptions and variations-making the task of data integrity enforcement a challenging one.

An integrated pathways system potentially incorporates a very large number of different types of biological information, e.g., about genes and proteins, such information include synonyms, gene and protein sequence information, genetic map location, polymorphisms, transcription factors, expression patterns (RNA, proteins), biochemistry, consequences of pharmacological perturbations, phenotypes, evolution, etc. Building all of these information in a pathway system is impossibly complex and unnecessary; a better alternative is to extract this information offline or online from the web-accessible resources. Ultimately, for difficult to extract data, one may envision the use of data extraction systems specialized for the tasks at hand. Finally, there is always the seamless data integration issue of the data available in the local database with the data in remote resources, on different components and in different levels.

8 Future Work

Our Pathways Database System is a typical computer science application for structured databases, involving querying and updating using views, XML views, data integration, query optimization. And, there are several graph traversals, path queries, for which special query evaluation techniques have potential to improve efficiency and need to be investigated.

An important goal is the integration of the three classes of pathways so that the information on different classes of pathways can be queried and visualized over the web in an integrated manner. Our long-range goal is to use the metabolic/genomic Pathways Database System for a number of (high-level) problems, including the relation between genetic mutations and functional consequences (such as metabolic dysfunction), and predictions of pathway composition and function.

Acknowledgments. Since its conception four years ago, a number of people contributed and helped develop Pathways Database System, including F. Akgul, A. Cakmak, M. Kirac, T. Kitami, L. Krishnamurthy, S. Newman, Y. Mei, C. Parker, N.Ratprasartporn, G. Schaeffer, W. Xu, B. Zhang. We also acknowledge Yoh-Han Pao for his continued support. This research has been funded by a gift from the Charles Wang Foundation and the NSF research grant DBI-0218061.

References

[K+a03] Krishnamurty, L., et al, *Pathways Database System: an Integrated System for Biological Pathways* Bioinformatics, Vol. 19, No. 8, 2003.

[K03] Krishnamurty, L., *Pathways Browser* M.S.Thesis, CWRU, 2003.

[M04] Mei, Y., *Pathway Viewer Tool* Version 1.0, Tech. Rep., CWRU, July 2004.

[NO04] Newman, S., Ozsoyoglu, M., *A Tree-Structured Query Interface for Querying Semi-Structured Data*, SSDBM Conf., June 2004.

[PSR04] Pathway System Release #2, available at http://nashua.cwru.edu/Pathways, 2004.

[PSRJ03] Java-based Pathway System Release #1, available at http://nashua.cwru.edu/Pathways, 2004.

[R03] Ratprasartporn, N., *J-Pathway Browser and Viewer Tool* Version 1.0, CWRU, July 2003.

[T04] Tasan, M., *Pathways Database User Guide*, Version 2.0, Tech. Rep., CWRU, July 2004

[X04] Xu, W., *Pathway Querying Services for Genome Pathways Database*, CWRU, July 2004.

Graph Data Representation in Oracle Database 10*g*: Case Studies in Life Sciences

Susie M. Stephens
Oracle Corporation
10 Van de Graaff Drive
Burlington
MA 01803
USA
susie.stephens@oracle.com

Johan Rung
McGill University and
Genome Québec Innovation Centre
740 Doctor Penfield Ave.
Montreal, QC, H3A 1A4
Canada
johan.rung@mail.mcgill.ca

Xavier Lopez
Oracle Corporation
One Oracle Drive
Nashua
NH 03062
USA
xavier.lopez@oracle.com

Abstract

*New technologies have been developed in the life sciences that allow researchers to study biological systems in rich detail. These advances have resulted in an abundance of data that describes the relations between the fundamental components of biological systems, such as genes, proteins, and metabolites. The network of relations between the components holds insights as to how biological systems function, and consequently can help researchers understand the mechanisms behind disease. Biological networks are commonly managed and analyzed in a graph representation. Oracle Database 10*g* has the functionality to model data as a graph, and thereby has the potential to greatly facilitate research. In this paper we describe the Oracle implementation and provide case studies from the life sciences.*

1 Introduction

Graph theory is an established field where users represent data as a series of nodes and edges [Pearl88]. A node represents an object of interest and an edge indicates a relationship between two nodes. Edges can have a direction indicating an ordering between the nodes, or be undirected. A graph typically assumes very little about the data it describes, enabling the nodes and edges of a graph to form a web of interconnectivity.

The behavior of systems is influenced by the topology of the graph, which represents the nature of the connections within the graph. Much early work in graph theory dealt with the properties of random graphs, where it is assumed that connections between any two nodes in a network are equally probable [ER60]. Many natural and man-made systems exhibit a scale-free topology where most nodes have just a few connections and only a few nodes form highly connected hubs [BA99]. Scale-free networks are considered to be robust as the risk of a random error to occur in a highly connected node is low. Such graphs also display a tendency to cluster, have shorter average path lengths between nodes, and are better able to demonstrate sustained growth [AB04, Sear03]. Graphs have been extensively investigated, for example in the social [Mil67], man-made [Al99] and biological domains [BO04, Alon].

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

For advances to be made in the understanding of biological systems and disease, it is critical for researchers to be able to gain insights into the role of different biological entities and their interactions. These interactions guide the life of a cell, allowing it to function and to respond to environmental stimuli. The size and complexity of biological systems makes analyzing their interactions challenging. Most cellular processes are a result of a cascade of events mediated by proteins. For example, a gene may express a transcription factor that regulates the expression of a different set of genes, a gene may express an enzyme that activates a set of proteins, two proteins may bind to each other to form a functional complex, or a gene may express an enzyme which catalyzes the production of a metabolic compound which in turn inhibits another enzyme.

Analysis algorithms such as nearest neighbor, minimum cost spanning tree, and within distance are commonly used with graphs to help researchers identify new insights in the data. These analytical techniques are especially powerful when combined with methods that enable cost or weight values to be assigned to edges, as this allows researchers to quickly limit a search to the desired targets. Cost values are typically used to represent interaction strengths, or to label an edge with a statistical value. The restriction of a graph to a sub-network of interest is critical as it enables graph visualization to become more manageable, and it reduces the computational complexity of subsequent analysis.

Managing life sciences data as a graph can facilitate the modeling of complex interactions because the emphasis is placed on the relationship between the objects. Graphs can enable complex networks to be visualized in a straightforward manner that captures the essential information about the structure of the system. Further, graphs that support hierarchies of information, where a set of nodes and edges can be contained within a parent node, are well suited to modeling the different organizational levels of biological. As a consequence of the suitability of graph modeling to life sciences data, its popularity as a method for data representation has increased over recent years. The range of applications of graph theory have also grown from primarily being used for metabolic pathway representation [Jeong], to areas as diverse as protein-protein interaction networks [Gag04, Reiss, Betel, Uetz], analyzing cell images [Gunduz], gene function determination [Sch03] and text mining [Je01].

Graphs have also been used in combination with fixed vocabularies and ontologies to aid scientists in answering questions of a more open-ended nature. For example, if protein data is represented as a graph and is mapped to an ontology that contains a concept such as INTERACTS-WITH, it becomes possible to identify all protein-protein interactions. A graph data representation can enable scientists to discover new insights by using pre-determined rules and ontologies to traverse across data sets.

The increasing number of data sources that contain biological pathway and interaction data, such as KEGG [Kan97, Kan00], BIND [BIND03], IntAct [IntAct04], DIP [DIP00], MINT [MINT02] and Ecocyc [Karp00], demonstrate the importance of systems data to the life sciences. As these data sources become more prevalent, and increase in size and complexity, it becomes more important that these data sources can be managed as a graph representation in a relational database management system (RDBMS). A RDBMS offers users the ability to store data in a secure, highly available and scalable environment. To date, it appears Oracle is the only relational database vendor to offer graph functionality in the database. This paper focuses on the implementation of a graph data model in Oracle Database 10*g*, and provides examples of its use within the field of bioinformatics.

2 Implementation

Oracle Database 10*g* includes a Network Data Model (NDM) as part of Oracle Spatial that enables users to model and analyze data as a graph [OraNDM]. NDM stores objects of interest as nodes, the relationship between nodes as links, and an ordered list of links that contain no repeating links or nodes as paths. NDM can be used to represent directed, undirected, random, scale free and hierarchical graphs. In addition, NDM has the ability to support logical graphs, and graphs that relate to spatial information. NDM is a model that represents networks in object-relational form in the database and as Java objects in the client or application tier.

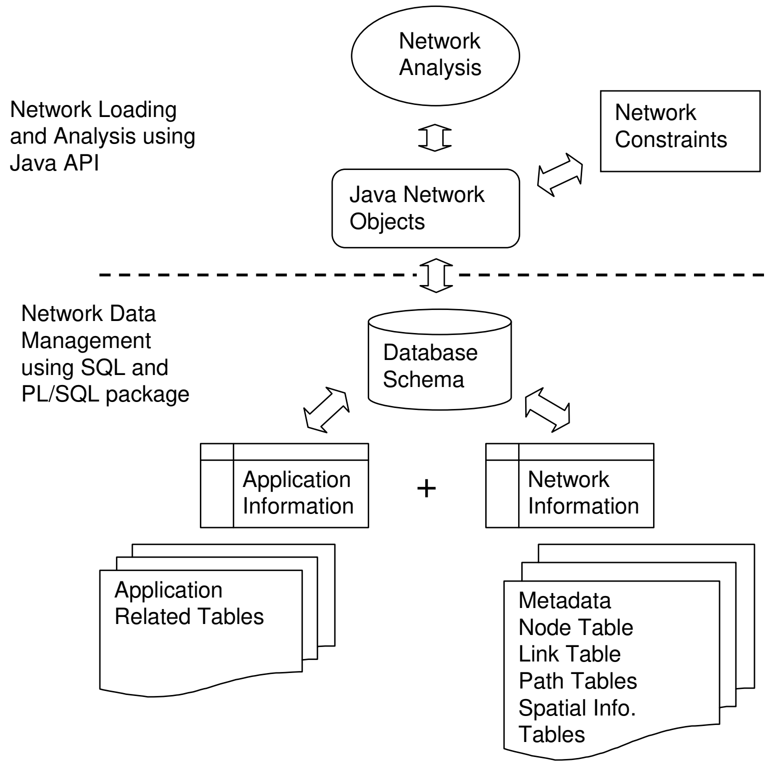


Figure 1: The Architecture of Oracle Spatial Network Data Model.

The NDM schema contains tables for storing the node, link and path data, and the network metadata. The NDM schema also includes a PL/SQL package for querying network data, maintaining referential integrity, and updating link, node and path tables. Standard SQL queries and PL/SQL functions and procedures can be performed on the data. Only generic connectivity information is captured in the data model, thereby providing a clean separation between the data and application information.

The Java API enables network loading, network representation and network analysis. NDM provides analytical capabilities for tracing, shortest path, minimum cost spanning tree, within cost and nearest neighbors. NDM also supports network constraints that are useful filters for network analysis, including minimum bounding rectangle, path cost, path depth, and must avoid nodes and links.

3 Network Data Model Case Studies

In the life sciences there are many possible use cases for a graph data model. In this section two possible use cases will be highlighted: metabolic pathway analysis and the integration of gene networks with gene expression data.

3.1 Metabolic Pathway Analysis

A graph data model is well suited to storing and analyzing metabolic pathways. Data from the KEGG metabolic pathway database from the University of Kyoto was loaded into NDM as a directed network. The chemical compounds were stored as nodes (table 1) and the enzymes were stored as links (table 2). Once the graph was

Table 3: Pathway Data Definition: Node table

NODE_ID	NODE_NAME	ACT	COSTS	SAMPLE_ID	ENTRY_ID
1	C00022	Y	1	Pyruvate	49
2	C00122	Y	1	Fumarate	50
3	C00036	Y	1	Oxaloacetate	51
4	C05379	Y	1	Oxalosuccinate	52
5	C00074	Y	1	Phosphoenolpyruvate	53
6	C00024	Y	1	Acetyl-CoA	54
7	C00149	Y	1	(S)-Malate	55
8	C00311	Y	1	Isocitrate	56
9	C00417	Y	1	cis-Aconitate	57
10	C00042	Y	1	Succinate	58

Table 4: Pathway Data Definition Link table

LINK_ID	LINK_NAME	START_NODE_ID	END_NODE_ID	ACT	COST	SAMPLE_ID
1	1.1.1.42 (rn:R00268)	4	19	Y	1	isocitrate dehydrogenase (NADP)
2	1.1.1.42 (rn:R00268)	19	4	Y	1	isocitrate dehydrogenase (NADP)
3	1.1.1.42 (rn:R00268)	4	20	Y	1	isocitrate dehydrogenase (NADP)
4	1.1.1.42 (rn:R00268)	20	4	Y	1	isocitrate dehydrogenase (NADP)
5	4.1.1.49 (rn:R00341)	3	19	Y	1	phosphoenolpyruvate carboxykinase (ATP)
6	4.1.1.49 (rn:R00341)	3	5	Y	1	phosphoenolpyruvate carboxykinase (ATP)
7	1.1.1.37 (rn:R00342)	3	7	Y	1	malate dehydrogenase
8	6.4.1.1 (rn:R00344)	1	3	Y	1	pyruvate carboxylase

stored in NDM, a Java visualizer was used to view the metabolic pathways and to interrogate the data using analysis including shortest path and all paths. Metabolic pathway analysis can guide researchers as to which drug discovery targets are likely to yield the most beneficial results.

3.2 Gene Network Analysis

Gene networks describe relations within a group of genes. Common approaches to gene network analysis include using the edge to represent correlations between gene expression profiles [Sz99], high mutual information [Butte], or the impact of a gene mutation on gene expression [Rung]. More complex examples include using a mathematical framework of probabilistic graphical models, such as Bayesian networks, to learn how the structure and parameters of a network fit observed data [Fr04, Hec95]. Gene networks can also be used to describe interactions between transcription factors and genes, where an edge typically represents a transcription factor physically binding to a regulatory site in the vicinity of a gene [Lee02].

This case study describes how gene expression data can be mapped to a signaling pathway, thereby providing additional insights to the perturbed component of the biological system. Gene expression profiles are of significant interest in life sciences as they can provide valuable insights into the underlying mechanisms of disease [Ra02]. In this example, signaling pathways are stored in NDM with genes represented as the nodes and interactions between the genes as the edges. Gene expression data from biological samples is stored in a separate relational table. Queries can be performed that span both the signaling pathway and the gene expression data, enabling researchers to identify subsets of the signaling pathway that are likely to be impacted by the changes in the gene expression profile (fig. 2). With NDM it becomes simple to retrieve connectivity information about genes.

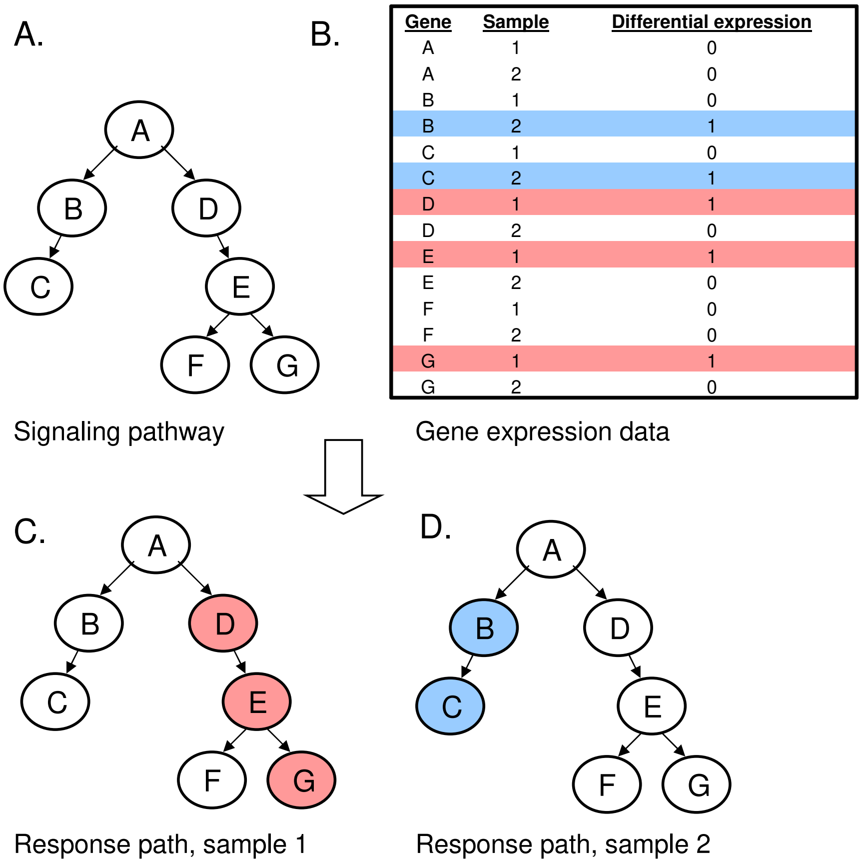


Figure 2: Case study for gene network analysis. A depicts a signaling pathway stored in NDM. B shows gene expression data stored in a relational table. C and D highlights the differential pathway expression between sample 1 (red) and sample 2 (blue) that was determined by performing a query that spans both data sets.

4 Conclusions

NDM enables scientists to manage and analyze data as a graph representation in Oracle Database 10*g*. NDM has been implemented by Oracle as an open and generic model that separates application logic from data management. This approach enables industries ranging from telecommunications, geographic information systems, electronics and life sciences to take advantage of the functionality.

In this paper, case studies were provided as to how NDM can be used for storage and analysis of metabolic pathways and gene networks. As increasing volumes of data in the life sciences are represented as a graph, it becomes important that a secure, reliable and scalable environment is provided for the data. As relational databases already manage much data in the life sciences, they should provide a strong platform for graph data management in drug discovery.

5 Acknowledgements

Data for table 1 and 2 used with the kind permission of Prof. Susumu Goto.

References

[Al99] R. Albert *et al.* Diameter of the World-Wide Web. *Nature*, 401: 130-131, 1999.

[AB04] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74: 47-97, 2002.

[Alon] U. Alon. Biological Networks: The Tinkerer as an Engineer. *Science*, 301: 1866-1867, 2003.

[BIND03] G.D. Bader *et al.* BIND: the Biomolecular Interaction Network Database. *Nucleic Acids Res.*, 31: 248-50, 2003.

[BA99] A.-L. Barabási and R. Albert. Emergency of Scaling in Random Networks, *Science*, 286: 509-512, 1999.

[BO04] A.-L. Barabási and Z.N. Oltvai. Network Biology: Understanding the Cells Functional Organization, *Nature Rev. Gen.*, 5: 101-113, 2004.

[Betel] D. Betel *et al.* Analysis of domain correlations in yeast protein complexes. *Bioinformatics*, 20 Suppl. 1: 55-62, 2004.

[Butte] A.J. Butte *et al.* Discovering Functional Relationships Between RNA Expression and Chemotherapeutic Susceptibility using Relevance Networks, *Proc. Natl. Acad. Sci. USA*, 97: 12182-12186, 2000.

[ER60] P. Erdös and A. Rényi. On the evolution of Random Graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5: 17-61, 1960.

[Fr04] N. Friedman. Inferring Cellular Networks Using Probabilistic Graphical Models, *Science*, 303: 799-805, 2004.

[Gag04] J. Gagneur *et al.* Modular decomposition of protein-protein interaction networks. *Genome Biology*, 5:R57, 2004.

[Gunduz] C. Gunduz *et al.* The cell graphs of cancer. *Bioinformatics*, 20 Suppl. 1: 145-151, 2004.

[Hec95] D. Heckerman *et al.*. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data, *Machine Learning*, 20: 197-243, 1995.

[IntAct04] H. Hermjakob *et al.* IntAct - an open source molecular interaction database. *Nucleic Acids Res.*, 32: D452-D455, 2004.

[Je01] T.-K. Jenssen *et al.* A Literature Network of Human Genes for High-throughput Analysis of Gene Expression. *Nature Gen.*, 28: 21-28, 2001.

[Jeong] H. Jeong *et al.* The large-scale organization of metabolic networks. *Nature*, 407: 651-654, 2000.

[Kan97] M. Kanehisa. A database for post-genome analysis. *Trends Genet.*, 13: 375-376, 1997.

[Kan00] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.*, 28: 27-30, 2000.

[Karp00] P.D. Karp *et al.* The ecocyc and metacyc databases. *Nucleic Acids Res.*, 28: 56-59, 2000.

[Lee02] T.O. Lee *et al.* Transcriptional Regulatory Networks in Saccharomyces cerevisiae. *Science*, 298: 799-804, 2002.

[Mil67] S. Milgram. The small-world problem. *Psychology Today*, 1: 60-67, 1967.

[OraNDM] http://www.oracle.com/technology/products/spatial/pdf/10g_network_model_twp.pdf

[Pearl88] J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, (Morgan Kaufmann Publishers), 1988.

[Ra02] S. Ramaswamy and T.R. Golub. DNA Microarrays in Clinical Oncology, *J. Clin. Oncology*, 20: 1932-1941, 2002.

[Reiss] D.J. Reiss and B. Schwikowski. Predicting protein-peptide interactions via a network-based motif sampler. *Bioinformatics*, 20 Suppl. 1: 274-282, 2004.

[Rung] J. Rung *et al.* Building and Analyzing Genome-wide Disruption Networks. *Bioinformatics,* 18 Supp.2: 202-210, 2002.

[Sch03] T. Schlitt *et al.* From Gene Networks to Gene Function. *Genome Research*, 13: 2568-2576, 2003.

[Sear03] D. Searls. Data integration - connecting the dots. *Nature Biotech.*, 21: 844-845, 2003.

[Sz99] Z. Szallasi. Genetic Network Analysis in light of massively parallel biological data acquisition. *Pacific Symposium on Biocomputing*, 4: 5-16, 1999.

[Uetz] P. Uetz *et al.* A Comprehensive Analysis of Protein-Protein Interactions in Saccharomyces cerevisiae. *Nature*, 406: 623-627, 2000.

[DIP00] I. Xenarios *et al.* DIP: The Database of Interacting Proteins. *Nucleic Acids Res.*, 28: 289-91, 2000.

[MINT02] A. Zanzoni *et al.* MINT: a Molecular INTeraction database. *FEBS Letters*, 513: 135-140, 2002.

ICDE 2005 TOKYO
International Conference on Data Engineering

The 21st International Conference on

Data Engineering (ICDE 2005)

April 3-9, 2005
National Center of Sciences, Tokyo, Japan
Sponsored by
The IEEE Computer Society
The Database Society of Japan (DBSJ)
Information Processing Society of Japan (IPSJ)
The Institute of Electronics, Information and Communication Engineers (IEICE)

http://icde2005.is.tsukuba.ac.jp/ http://icde2005.naist.jp/ (mirror)

Welcome to ICDE 2005 Tokyo

Data Engineering deals with the use of engineering techniques and methodologies in the design, development and assessment of information systems for different computing platforms and application environments. The 21st International Conference on Data Engineering (ICDE 2005) provides a premier forum for:

- sharing research solutions to problems of today's information society
- exposing practicing engineers to evolving research, tools, and practices and providing them with an early opportunity to evaluate these
- raising awareness in the research community of the problems of practical applications of data engineering
- promoting the exchange of data engineering technologies and experience among researchers and practicing engineers
- identifying new issues and directions for future research and development work.

The conference will be held in Tokyo, the capital city of Japan. Tokyo is the most populous metropolitan area in the world. Yet despite its complex urban landscape and impressive architecture, this city abounds with parks and gardens. Particularly in the spring season, blooming *cherry blossoms* will welcome you to Tokyo.

Advanced Technology Seminars

- XQuery Midflight: Emerging Database-Oriented Paradigms and a Classification of Research Advances (I. Manolescu and Y. Papakonstantinou)
- Data Stream Query Processing (N. Koudas and D. Srivastava)
- Web Service Coordination and Emerging Standards (F. Casati and G. Alonso)
- Online Mining Data Streams: Problems, Applications, Techniques and Progress (H. Wang, J. Pei, and P. Yu)
- Rank-Aware Query Processing and Optimization (I. Ilyas and W. Aref)
- Data Mining Techniques for Microarray Datasets (L. Liu, J. Yang, and A. Tung)

Workshops

- International Workshop on Ubiquitous Data Management (UDM 2005), Apr. 4, 2005. Deadline: Nov. 15, 2004.
- International Workshop on Biomedical Data Engineering (BMDE 2005), Apr. 3-4, 2005. Deadline: Nov. 15, 2004.
- International Workshop on Challenges in Web Information Retrieval and Integration (WIRI 2005), Apr. 8-9, 2005. Deadline: Nov. 15, 2004.
- International Workshop on Privacy Data Management (PDM 2005), Apr. 8-9, 2005. Deadline: Nov. 17, 2004.
- International Workshop on Autonomic Database Systems, Apr. 8-9, 2005. Deadline: Nov. 15, 2004.
- International Workshop on Realworld Multimedia Corpora in Mobile Environment (RWCinME 2005), Apr. 8-9, 2005. Deadline: Nov. 15, 2004.
- 2nd International Workshop on XML Schema and Data Management (XSDM 2005), Apr. 8-9, 2005. Abstract Deadline: Oct. 15, 2004. Paper Deadline: Oct. 22, 2004.
- International Workshop on Data Engineering Issues in E-Commerce (DEEC 2005), Apr. 9, 2005. Deadline: Nov. 22, 2004.
- International Workshop on Managing Data for Emerging Multimedia Applications (EMMA 2005), Apr. 8-9, 2005. Deadline: Nov. 15, 2004.

Keynote Speakers (other speaker to be announced)

- Pat Selinger (IBM, USA)
- Mike Stonebraker (MIT, USA)

Conference Officers

Honorary Chair:	The Late Yahiko Kambayashi (Kyoto University, Japan)
Organizing Committee Chair:	Yoshifumi Masunaga (Ochanomizu University, Japan)
General Chairs:	Rakesh Agrawal (IBM Almaden Research Center, USA)
	Masaru Kitsuregawa (University of Tokyo, Japan)
Program Chairs:	Karl Aberer (EPF Lausanne, Switzerland)
	Michael Franklin (UC Berkeley, USA)
	Shojiro Nishio (Osaka University, Japan)
PC Area Chairs:	Anastassia Ailamaki (Carnegie Mellon University, USA)
	Gustavo Alonso (ETH Zurich, Switzerland)
	Phillip Gibbons (Intel Research, USA)
	Takahiro Hara (Osaka University, Japan)
	Jayant R. Haritsa (IISc Bangalore, India)
	Alfons Kemper (University of Passau, Germany)
	Sharad Mehrotra (UC Irvine, USA)
	Wolfgang Nejdl (University of Hannover, Germany)
	Jignesh M. Patel (University of Michigan, USA)
	Evaggelia Pitoura (University of Ioannina, Greece)
	Jayavel Shanmugasundaram (Cornell University, USA)
	Kyuseok Shim (Seoul National University, Korea)
	Kian-Lee Tan (National University of Singapore, Singapore)
Executive Committee Chair:	Jun Adachi (National Institute of Informatics, Japan)
Panel Chairs:	Umeshwar Dayal (HP Labs, USA)
	Hongjun Lu (HKUST, China)
	Hans-Jörg Schek (UMIT, Austria/ETH Zurich, Switzerland)
Seminar Chairs:	Michael J. Carey (BEA Systems, USA)
	Stefano Ceri (Politecnico di Milano, Italy)
	Kyu-Young Whang (KAIST, Korea)
Demo Chairs:	Daniel Keim (University of Konstanz, Germany)
	Ling Liu (Georgia Institute of Technology, USA)
	Xiaofang Zhou (University of Queensland, Australia)
Industrial Chairs:	Anand Deshpande (Persistent Systems, India)
	Anant Jhingran (IBM Silicon Valley Lab, USA)
	Yasushi Kiyoki (Keio University, Japan)
	Eric Simon (Medience, France)
Local Arrangement Chair:	Haruo Yokota (Tokyo Institute of Technology, Japan)
Workshop Chair:	Masatoshi Yoshikawa (Nagoya University, Japan)
Proceedings Chair:	Motomichi Toyama (Keio University, Japan)
Publicity Chair:	Hiroyuki Kitagawa (University of Tsukuba, Japan)
DBSJ Liaison:	Hiroshi Ishikawa (Tokyo Metropolitan University, Japan)
CCF-DBS Liaison:	Xiaofeng Meng (Renmin University, China)
KISS SIGDB Liaison:	Chin-Wan Chung (KAIST, Korea)
DBIndia Liaison:	Krithi Ramamritham (IIT Bombay, India)
Australian DB Liaison:	James Bailey (University of Melbourne, Australia)
Treasurer:	Miyuki Nakano (University of Tokyo, Japan)

Related Conferences and Workshops

- 15th International Workshop on Research Issues on Data Engineering: Stream Data Mining and Applications (RIDE-SDMA), Apr. 3-4, Tokyo.
- The Seventh Asia Pacific Web Conference (APWeb 2005), Mar. 29-Apr. 1, 2005, Shanghai, China. http://apweb05.csm.vu.edu.au/
- 10th International Conference on Database Systems for Advanced Applications (DASFAA 2005), Apr. 18-20, Beijing, China. http://dasfaa05.cs.tsinghua.edu.cn/
- 4th International Workshop on Databases in Networked Information Systems (DNIS 2005), Mar. 28-30, Aizu, Japan. http://www.u-aizu.ac.jp/labs/sw-db/DNIS2005.html

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398