Database Technology Summaries

Deductive Database System

A deductive database uses a declarative language, one that describes what you want from the data rather than how you want to get it, to outline rules. A “deduction mechanism” (Elmasri & Navathe, 2003) can then infer new facts about the dataset using the outlined rules. The deductive database uses Datalog (a variation of Prolog) to define rules to be applied to relations. The Datalog language is similar to SQL in that it forms an if-then structure however Datalog allows rules to be recursive.

According to (Ramakrishnan & Gehrke, 2003) deductive databases were a solution to the limitations in the query language and constraint definition capabilities of relational database systems, confirming this Alexander P. Pons states that in the mid-1990s researchers proposed deductive database systems to overcome the obstacles found in relational database systems (Pons, 2003).

Due to its ability to handle large amounts of data and apply reasoning and inferences a deductive database is

“Attractive to Decision Support that require a business’s historical and current information to make predictions” (Pons, 2003)

This would imply that deductive databases could be useful in a number of spheres e.g. Customer Relational Management systems, where predictions based on historical information would prove to be highly valuable to future sales and customer relationships. In a 2011 article Andreas Behrend states that until recently the deductive database inference methods had rarely been used in commercial database systems due to a lack of

“Uniform approach well-suited for implementation in an SQL-based system” (Behrend, 2011)

A company named HighFleet have been using deductive databases to provide their clients with

“Question-answering capabilities never before available”

And comments that

“Enterprises can understand what their data means without an excrescence of heavy weight, brittle and expensive applications or without weeks to months of special effort by IT staff” (HighFleet, n.d.)
In summary deductive databases are inference mechanisms that use rules and facts to give meaning to data. Although we do see deductive databases commercially they do not compare to the strong hold of relational databases.

**Parallel Database System**

Moving away from a centralised database management system, parallel databases are designed to provide a huge increase in performance through parallel command execution. In line with this Ramakrishnan & Gehrke recount that although data may be stored in a distributed fashion in such a system, the distribution is governed solely by performance considerations (Ramakrishnan & Gehrke, 2003). It should be noted that parallel databases are not equivalent to distributed databases, when discussing the distinction between parallel and distributed databases Özsu and Valduriez explain

“*Distributed DBMSs assume loose interconnection between processors that have their own operating systems and operate independently. Parallel DBMSs exploit recent multiprocessor computer architectures in order to build high-performance and high availability database servers at a much lower price than equivalent mainframe computer*” (Özsu & Valduriez, 1996)

There are three main architectures that have been proposed for parallel database management systems (DeWitt & Gray, 1992) (Ramakrishnan & Gehrke, 2003) (Mohamed et al., 1998) :

- **Shared Memory** - Any CPU has access to both memory and disk through a fast interconnect (e.g high-speed bus). This provides excellent load balance however scalability and availability is limited.

- **Shared Disk** - This provides the CPU with its own memory but a shared disk. Meaning there is no longer competition for shared memory but still competition for access to the shared disk. This provides better scale up and the load balancing is still acceptable. Availability is better than shared memory but still limited as disk failure would mean entire system failure.

- **Share Nothing** - Each processor has exclusive access to its main memory and disk unit, this means all communication between CPUs is through a network connection. Shared nothing has high availability and reliability; if one node fails the others are still able to run independently. However load balance and skew become major issues with this architecture.
The architecture believed to scale best is shared nothing (Abouzeid et al., 2009) and many of the large DBMS vendors such as Teradata and Microsoft use it.

```
SELECT SUM(REVENUE) FROM LINE_ITEMS;
```

When processing queries there are two different dataflow approaches a parallel database system could take:

- Pipelined parallelism – Pipelining is when a process is split into stages and those stages are run on multiple processors. When executing a command it first passes through to
stage one and when that has finished it passes off to the next stage and can pull a new command off of the queue and start executing at stage one.

- Partition parallelism – Partitioning means splitting the input data among multiple processors and memories, executing and then merging the outputs together.

![Diagram of dataflow approach](image)

**Figure 3** - The dataflow approach to relational operators gives both pipelined and partitioned (DeWitt & Gray, 1992)

Typical usage of a parallel database would be anywhere where performance needs to be high.

**Spatial Database System**

Put simply spatial databases are concerned with storing data in relation to space. Güting describes a spatial database as one that supports spatial data types in its data model. It provides a query language and at the minimum spatial indexing and spatial join methods. He goes on to state,

“*Spatial database systems offer the underlying database technology for geographic information systems and other applications*” (Güting, 1994)

Spatial databases were designed to simplify the querying of spatial data. Shekhar and Chawla (Shekhar & Chawla, 2003) explain

“a query like “List the top ten customers, in terms of sales, in the year 1998” will be very efficiently answered by a DBMS even if the database has to scan through a very large customer database”

and further clarify

“a relatively simple query such as “List all the customers who reside within fifty miles of the company headquarters” will confound the database”
This is caused by a lack of indexing for narrowing the search due to traditional indices being incapable of ordering multidimensional coordinate data. Thus the need for spatial databases to combat this issue was conceived.

There are a number of uses for spatial databases Elmasri and Navathe (Elmasri & Navathe, 2003) give cartographic databases as one example. A cartographical database is one that contains coordinates defining a geographical area which can then be used with other variables (such as age, gender or class) to map the distribution in a geographical area. Another example of spatial databases in use is one taken from a presentation by Bettina Berendt, Figure 4 shows a customer subscribing to be notified of a shopping coupon, should they be in the location of a shop that provides one they will be alerted. This obviously provides a clear financial benefit for consumer and producer; the consumer gets a discount and the provider makes a sale to a customer that may otherwise have not known about them.

![Diagram of location-based services](image)

**Figure 4 - Customer being sent shopping coupon (Berendt, 2007)**

**Temporal Database System**

Temporal databases are concerned with storing data relating to time instances. It provides temporal data times and stores data relating to past, present and future time. There are a number of examples of applications where temporal database technology is used. Financial applications such as; FX trading platforms, accounts and banking and also scheduling applications such as; hotel reservations, project management and train schedules. (Jensen, 2000)
A temporal database is an ordered sequence of time duration points set by the application known as chronon. Chronon describe the minimal granularity this maybe be 1 millisecond, 1 second or 1 minute, whichever the application chooses. However choosing a minimum granularity can lead to events occurring within the same period, they would then be considered simultaneous events when this may not be the case.

There are two common aspects that are associated in the temporal database; they are referred to as time dimensions (Wikipedia, 2011) (Elmasri & Navathe, 2003):

- **Valid time** – A time period in which a fact is true or false in the “mini-world”, a database representation of the real world. A temporal database using this is called a valid time database.

- **Transaction time** – A time period when the fact is stored in the database. The fact is rarely deleted, as it may need to be used for auditing purposes but is considered to be invalid. A temporal database using this is called a transaction time database.

Applications can decide to make use of either type or a combination of both known as a bitemporal database. If neither of these options proves to be appropriate for the application a custom user defined solution can be created and is called a user-defined time database.

**Parallel and Spatial Databases: Further Analysis**

**Parallel Database System**

Parallel database technology has been around for over 25 years. Although initially the forecast for parallel databases looked meek (DeWitt & Gray, 1992) in recent years we have seen them enter main stream. This could be attributed to the rise and domination of the relational model and the suitability of relational queries for parallel execution. The aim has always been to improve performance and the problem faced by conventional database management was known as “I/O bottle neck” (Valduriez, 1993) which was caused by high disk and memory access time. Initial solutions saw mainframe designers use special-purpose hardware, however this failed due to high costs and poor performance. Meanwhile fast and inexpensive multi-processors began to become widely available that would make machines more powerful and cheaper than their mainframe counterparts. Parallel databases have exploited the move to fast and cheap disks, processors and memory and combined it with the popularity of the relational model to establish itself commercially.
With the introduction of parallel databases, users have had to adjust the design of their applications and databases to allow for a seamless experience. Oracle state that there are four levels of scalability needed for the successful implementation of parallel processing and parallel databases (Oracle, n.d.):

- **Hardware** – Every system must be able to access the CPUs, since they are at the core of parallel execution. Bandwidth and latency of the access link determine the scalability of the hardware.

  ![Interconnection Network](image)

  **Figure 5 - Example of interconnection network for Shared-nothing Architecture**

- **Operating System** – This is an important issue if there is a shared memory node. Synchronization done by the operating systems can impact the scalability of the whole system. For example if the OS was developed for single CPU systems you would have asymmetric multiprocessing this is where only a single CPU can handle I/O interrupts. If that system has multiple user processes request a resource from the OS then you would create a bottle neck and hardware scalability is lost as a result. This means you need to make sure that the OS is Symmetric multiprocessing, most of today’s OS are of that kind, this problem would only be highlighted in legacy systems.

  ![Asymmetric multiprocessing](image)

  **Figure 6 - Asymmetric multiprocessing**
Database Management System – It is important to decide on whether the parallelism takes part internal or externally. That is, does the DBMS parallelise the query or does an external process parallelise the query. Synchronisation is highly important and an efficient DBMS enables better speedup and scale up.

Application – Applications need to be designed to be scalable. Despite hardware, software and the database being scalable if the application, for example, a table with only one row which every node is updating this will synchronize on one data block.

Oracle gives the following example:

```sql
UPDATE ORDER_NUM
SET NEXT_ORDER_NUM = NEXT_ORDER_NUM + 1;
COMMIT;
```

**Code snippet 1 - Causing data block**

Code snippet 1 shows that every node that needs to update the order number has to wait for the row of the table to become free. A better solution would be:

```sql
INSERT INTO ORDERS VALUES
(order_sequence.nextval, ...)
```

**Code snippet 2 - more scalable solution to code snippet 1**

A consideration must also be made into how the clients are connected to the server machines; this should be done in a scalable manner which implies that the network must also be scalable.

In summary the points that application and DB designers have had to take into consideration are:

- **Synchronisation** – How do we ensure data integrity and prevent concurrency issues? Can the application provide “dirty data”? What type of locking does my application require?
- **Data separation/placement** – How will the data be separated?
- Query optimisation – How will the queries be optimised to allow for parallel execution?

After conducting research into emerging parallel database technologies it was discovered that companies are now often reporting to load more than a terabyte of structured data a day into their analytical database systems (Monash, 2008), this will most likely continue to rise and performance expectations remain if not increase. Companies are already finding that traditional parallel database solutions can be too expensive on massive datasets resulting in several big names developing distributed data storage and processing systems on large clusters of shared-nothing servers (Chaiken et al., 2008). These companies include Google’s File System, BigTable, MapReduce, Hadoop and Microsofts Dryad. These clusters can consist of hundreds or even thousands of machines and so writing a program that maximises the parallelism can prove to be difficult. There are now immerging abstraction frameworks such as MapReduce that are providing a more user friendly way of programming, however users have to map their applications to the MapReduce model to provide parallelism which renders the code coupled and less reusable. We are already beginning to see new languages, such as SCOPE, form in an attempt to resolve the problems. Over the next 5-10 years we might expect to see a move away from parallel databases (PDB) and the evolution of the already existing hybrid combination of PDB’s and distributed computing frameworks, such as MapReduce. Abouzeid et al. state in their paper:

“Ideally, the scalability advantages of MapReduce could be combined with the performance and efficiency advantages of parallel databases to achieve a hybrid system that is well suited for the analytical DBMS market and can handle the future demands of data intensive applications”

In 2011 there was an article written on the prototyping of such hybrids LinearDB (Wang et al., 2011). It describes the explosion of the scale of the data warehouse and the move away to “private clouds” based on clusters of low-cost machines. The paper reiterates that parallel databases architectures are not designed for several hundred inexpensive machines which are likely to be unreliable and also that a MapReduce based system, although more scalable, has inferior performance. This would further suggest a possibility of a lean towards hybrid systems.

AT&T are reported to use Teradata as a data warehouse provider (Wikipedia, n.d.). Since Teradata implement Parallel databases in their solutions it would indicate that AT&T are using them. Furthermore AT&T is an American multinational telecommunications corporation (Wikipedia, n.d.) this would imply they have huge amounts of data to process. Parallel
databases are a fitting solution to provide AT&T with fast processing of large amounts of data to provide their customers with a quick service.

**Spatial Database System**

The initial problem standard DBMSs had with handling spatial data was the implementation of spatial algebra and the integration of it into the querying process (Güting, 1994). The DBMS's were not designed to provide spatial data types and implementation of atomic operations, spatial indexing to support spatial selection and support of spatial join. A DBMS that wanted to support spatial data would have to accommodate the following in its architecture:

- representations for the data types of a spatial algebra,
- procedures for the atomic operations,
- spatial index structures,
- access operations for spatial indices,
- filter and refine techniques,
- spatial join algorithms,
- cost functions for all these operations,
- statistics for estimating selectivity of spatial selection and spatial join,
- extensions of the optimizer to map queries into the specialized query processing methods,
- spatial data types and operations within data definition and query language,
- user interface extensions to handle graphical representation and input of SDT value.

When relational systems were developed attempts were made to use them as a basis. There emerged two main architectures Layered Architecture, also called pure relational and Dual Architecture, also called loosely coupled.

![Layered architecture](image)

**Figure 8 - Layered architecture – Spatial functionality is implemented on top of underlying functionality**

To represent spatial data types (SDT) there were two strategies. The first strategy was to break the SDT down and place its values in tuples, the issue with this was that for an instance of the SDT to be used in would have to be reconstructed first which would be process heavy and thus
expensive. The second option was to represent the SDT values in “long fields” this was better but still came with problems as the DBMS would handle the geometrics only in the form of interpreted byte strings so any operation or evaluation on the actual geometry instance could only be done in the top layer.

Dual architecture brought in an integration layer. The top layer integrates the standard DBMS and the spatial subsystem.

![Dual Architecture](image)

**Figure 9 - Dual Architecture**

Here the SDT is broken down and non-spatial attributes is stored in the standard DBMS while the spatial attributes are stored in the spatial subsystem and the two pieces are connected by logical pointers.

The previous two architectures assumed that the DBMS was closed to extensibility. There is a third option which is the most popular and used by commercial companies such as PostgreSQL and Oracle (Browne, 2009). This is where the DBMS is extended to cater for spatial data. The query language is extended and new spatial types are handled as basic types by the DBMS. The disadvantage to this type of architecture is that you must use additional software to make an attractive map from the raw vectors stored within the DBMS.

Designers have had to approach spatial databases in a more formal manner. The design of a spatial database has a huge impact on the special querying, analysis, data exchange and system operability (Bédard, 1999). Spatial data is highly complex and such complexity requires high performance which encourages database structures that are hard to understand and so the analysis and design of special databases must rely on formal effective methods and patterns. Bédard explains that when designing for spatial databases there is a clear split in the form of analysis and the design and that this split is essential for multiplatform environments. Analysis design relates to the users perspective of the application, say in the form of roads and houses, whereas the design is more focused on the technology selected.

Object relational DBMSs are often used for handling spatial data due to their ability to handle complex objects (Browne, 2009). An object relational DBMS is strongly linked with object
orientated programming and like OOP. OOP can be engineered using unified modelling language, Bédard suggests a similar approach to designing for spacial data and hints that progress in reverse engineering tools would be an advantage.

Over the next 5-10 years we should also expect to see a more uniform cost efficient solution

Jayant Sharma Oracle Spatial’s technical director comments:

“While in the past, systems and applications focused on solving a single problem using custom data models, and specialized databases, enterprises are moving away from these isolated systems to reduce duplication of effort, reduce costs through better utilization of resources, and improve quality of service.”

Research appears to indicate towards the development of a more comprehensive and effective tool set for spatial data analysis (Sharma, 2005) which will make it easier for users to extract valid data. The GIS community also mention that ease of use is an aspect we can potentially look forward to in the future, with one member stating:

“We are hiding more of the underlying technology and making it easier for users to ask GIS questions, and make maps” (Snape, 2011)

In the same paragraph Snape does express that he views the future of spatial databases as divided and that while we are starting to make things easier we are also starting to look at other options for analysis. One such option was suggested to be Graph (NoSQL) databases for the storage and retrieval of spatial data.

One application that utilises a special database is the public land survey system (Wikipedia, n.d.) It is used in the United States to survey and spatially identify land parcels before designation of eventual ownership.
The information recorded is perfectly suited to spatial database technology. Storing the data in any other database would not be suitable because it would provide the querying and data types needed to represent meridians and baselines. The system presents to the user highly complex spatial data and the needs to be manipulated in a variety of ways that is only suited to a spatial database.

Cloud Computing, the Semantic Web and the Future of Database Technology

“Internet based technologies are likely to have a profound impact on how we view database systems in the future; especially as communications media is deployed supporting ever-increasing bandwidths. Two example technologies are the semantic web and cloud computing, both of which are already showing commercial promise.”

Considering the above statement I conducted research in two parts and summarised my argument below.

Cloud Computing

Cloud computing is essentially providing the capability to use computing and storage as a remote service. Nicholas Carr (Carr, 2008) comments that computing will be much like electricity: purchase when needed, referring to it as “utility computing”. That is that you pay a
company, such as Amazon, to provide you with the computing and storage you require and you keep paying them for as long as you need it. If you require more or less services then adjust your payments accordingly. Typically the provider will produce a Service Level Agreement (SLA) which they will adhere to in providing you that service. For example you may want to back up all of your home videos, photos and mp3’s in case of hardware failure or just due to the amount of space these files are taking up. Companies like humyo.com offer a cloud based service that will store all this data for you and at a minimal cost.

After researching the architecture of cloud computing (Appendix 1 – Cloud Computing Architecture) we know what cloud computing is. How can this affect the future of database technology? The Claremont report on Database research 2008 (Appendix 3 – Claremont Report) stated that there is a “turning point in the history of the field [databases]” and that this is due to the explosion of data and usage scenarios as well as huge shifts in computing hardware and platforms. A part of this turning point is listed as being cloud computing. Companies are continually making efforts to reduce computing costs (Han, 2010) and in today’s economic climate more so than ever. This is where cloud computing provides a solution that has seen large companies like Amazon and Yelp move to the cloud. Moving storage facilities offsite and into the cloud would save large amounts of money in maintenance and support, hardware, licencing and storage space, this is attractive to any company. Aside from the cost savings, services are available on request without waiting which means should you require extra storage it’s available instantly. If the services being used are on a grid computing system then companies would be able to take advantage of the entire networks processing power which aligns with the Claremont report statement of “Cloud computing democratises access to parallel clusters of computers”.

An example of a company moving to the cloud is 37 Signals. According to amazon web services case study (Amazon, n.d.) 37 signals were original using Network File System (NFS) server for storage. When their data began to grow towards 1 terabyte they needed an alternative solution. David Heinemeier Hansson, 37signals’ Partner, is reported to have said

“The cost and time associated with maintaining a 1 terabyte file server with full backups and zero downtime are significant when you’re living off managed hosting.”

37 Signals moved their products Basecamp, Highrise, Campfire, Backpack, Ta-da List, and Writeboard to the cloud stating
“I thought S3 was brilliant. I like low risk, outsourced services that give us room to grow with very little initial outlay”.

With all the positive vibe around cloud computing are there any drawbacks? The biggest concern that companies are having with cloud computing is security and privacy. This, for some corporations such as the government, is of up-most importance. Security concerns companies because they are not able to visibly see how secure their data is, however it has been suggested that the providers of cloud computing have to have reliable security measures in place otherwise they would lose all of their clients. Privacy is a different concern with users being able to access data from anywhere privacy is compromised. One way to combat this has been to enforce authorisation and authorisation groups, where users can only access certain resources. Industry professionals, Law firms and universities have also been debating more philosophical questions such as. Who owns the data, the client or the storage provider? Can clients be prevented from accessing their own data?

Although the Claremont report expresses that it anticipates future data centric applications will leverage data services in the cloud they do highlight that more work is needed to understand the scaling realities of cloud computing. However in a more recent report by Salesforce (Appendix 4 – Salesforce Report) they are quoted as saying:

“[cloud databases] they transparently scale according to varying application workloads”

In summary the cloud brings such large cost and time benefits that a number of companies will view its advantages to far outweigh the disadvantages. However in the database space DbaaS is a relatively new concept and issue with security, privacy and data ownership are likely to deter the highly security conscious companies. In the future we should expect to see proof of improvements in security and privacy as it is in the service providers’ best interests. In light of increased security and privacy we are likely to see larger security conscious companies’ move to using the cloud as a storage facility.

**The Semantic Web**

To understand the semantic web we must first understand that the web is a space of information currently used for human-human communication. The content on the web is currently unstructured, by that we mean that the structure is not evident to a robot browsing the web (Berners-Lee, 1998). The semantic web is about structuring content on the web in such a language that expresses information in a machine interpretive way this means we can expose the information hidden in text or blobs of media (Wolf, 2010).
The semantic web is a layered architecture (Hall & O'Hara, 2009) to be able to understand this architecture and how its implementation a number of terms must first be explained first, Appendix 2 – Semantic Web Architecture and Terminology.

In 2004 a book by Gutierrez et al. stated that:

“Research on formal aspects of RDF data and query languages is made necessary by the new features that arise in querying RDF graphs as opposed to standard databases”

Which clearly highlights that the semantic web will not be using database technology as we know it today, the traditional database does not translate directly to the RDF setting (Gutierrez et al., 2004). What we see in this chapter is references to an RDF database that can be considered a standard relational database with a more suitable querying system.

“An RDF graph can be considered a standard relational database: a relation of triples with the attributes Subject, Predicate, and Object. In what follows, an RDF database will be simply an RDF graph.”

Tim Berners-Lee appears in agreement with that statement, saying:

“The semantic web data model is very directly connected with the model of relational databases” (Berners-Lee, 1998)

In 2007 Feigenbaum et al. wrote an article detailing applications of the semantic web in today’s world. One case study is based around drug discovery and how semantic capabilities can be leveraged to find the underlying genetic causes of cardiovascular diseases. It describes how data was taken from databases in different departments of the hospital and translated into RDF format and stored in a “Semantic Web database”. Another example given is a case study in health care. A system called SAPPHIRE integrates a wide range of data from local health care providers, hospitals, environmental protection agencies and scientific literature. SAPPHIRE was configured to help with combating the spread of disease when public health officials became concerned after Hurricane Katrina.

“SAPPHIRE succeeded in identifying gastrointestinal, respiratory and conjunctivitis outbreaks in survivors of the disaster much sooner than would have been possible before.” (Feigenbaum et al., 2007)

With any emerging technology there are always issues. In an article about the semantic web (Rapoza, 2007) Rapoza notes that there will be vulnerabilities to scammers, which could lead
to security issues and therefore data protection problems, Rapoza gives the example of phishing sites and how, just like those, it could be possible to legitimate a source. There is also a potential issue with access, especially when reviewing the commercial aspects, Burners-Less had said that this is an area for focus for the Semantic Web community. One of the key adoption blocks that is named in the article by Rapoza is greed. Businesses are reluctant to expose their data and actively develop custom formats to keep people using their products and sites.

In summary the future of the semantic web is unclear; there are a number of hurdles that have to be overcome before we see this go main stream. It’s clear there is a need for a more structured “Web of Data”, especially with information on the web growing at an alarming rate. Research suggests that we will see more not for profit companies make a move towards utilising the semantic web before we see commercial businesses move and if they do the top tiers of the architecture will have to be clarified and standardised.

**Summary**

Research would indicate that the aforementioned technologies will have a large impact on the way we view database systems in the future. Database as a Service (DbaaS) is already a very real option for many companies, the cloud is already making its impact and we are likely to see this impact rapidly grow. The semantic web has proved to be of great use although adoption is not currently wide there is a definite need. Research already describes the impact RDF has on databases, as we move forward it is possible we will see a more specialised database or a highly functional hybrid.

At the time if this report and through analysis of the research it could be suggested that cloud technologies are going to leave bigger impact on database technologies than the semantic web over the next 5-10 years.
References


Appendix 1 – Cloud Computing Architecture

Cloud computing systems can be considered to have “front end” “back end” where the clients computer or network and application requesting resources from the cloud is considered the front end and the back end being the cloud which can consist of potentially any computer program, from data storage to video games. Architecturally the back end has a central server which monitors requests and utilizes servers through the use of middleware. Server utilization is maximised my virtualisation, convincing the physical server it is more than one.

Figure 11 - Basic architectural diagram of Cloud computing
Appendix 2 – Semantic Web Architecture and Terminology

Figure 12 - Semantic web architecture

URI – Unique resource identifier this gives a “resource” a unique identity. It contains a string of characters for identifying an abstract or physical object or resource. It is not a URL a URL is a uniform resource locator and is a type of URI that identifies a resource by its location on the network.

Unicode – An industry standard for consistent encoding, representation and handling of text

XML – Extensible Markup Language is a way for users to tag their data with descriptive custom tags. It is a basic language for representing and exchanging structure information. HTML is a well-known standard language for displaying content on the web.

RDF – Resource definition framework, is a way of representing data in “triples” of subject, predicate and object to enable a machine to understand and contextualise. It uses URIs to refer not only to the two items related, but also to the relationship asserted between them.

A Sentence

Jane

Sells

Books

Figure 13 - A Triple
By having this structure relationships can be formed, Figure 14 - RDF graph showing how collections of RDF statements create a "web of data" could be written in RDF/XML as a single document or in several that are linked together.

![RDF graph showing how collections of RDF statements create a "web of data" (Burleson, 2007)](image1)

Figure 14 - RDF graph showing how collections of RDF statements create a "web of data" (Burleson, 2007)

![Network of data sets. Colours are used to indicate different topics. The weights of links are represented through varied line width (GUÉRET et al., 2011).](image2)

Figure 15 - Network of data sets. Colours are used to indicate different topics. The weights of links are represented through varied line width (GUÉRET et al., 2011).
RDF-S – Is the schema for the RDF. It describes relationships which can be asserted between resources.

RIF – Rule exchange format, is a language to express the most common or basic types of rule. Rules allow transformation and inference of data.

OWL – Web Ontology Language for describing ontologies on the web.

SPARQL – Protocol And RDF Query Language. SPARQL is a special query language designed specifically for querying data stored in RDF.

Ontology – Ontology defines objects that exist in a domain of knowledge and the relationships between them. It is a contract that outlines the form of the actual instance of the knowledge.

As you can see in Figure 12 - Semantic web architecture and the development of these layers has been bottom-up. The bottom layers are more thoroughly defined and come with W3C recommendations whilst the upper layers are still the focus of research and clarification. Unifying logic, proof systems and trust is related to confirming the validity of the source of the data. If users are to infer facts from data then they will want to trust the sources. Little is mentioned in articles about these top layers so an assumption has been made that research work and clarification is on-going.
Appendix 3 - Claremont Report

The Claremont Report on Database Research


Abstract
In late May, 2008, a group of database researchers, architects, users and pundits met at the Claremont Resort in Berkeley, California to discuss the state of the research field and its impacts on practice. This was the seventh meeting of this sort in twenty years, and was distinguished by a broad consensus that we are at a turning point in the history of the field, due both to an explosion of data and usage scenarios, and to major shifts in computing hardware and platforms. Given these forces, we are at a time of opportunity for research impact, with an unusually large potential for influential results across computing, the sciences and society. This report details that discussion, and highlights the group’s consensus view of new focus areas, including new database engine architectures, declarative programming languages, the interplay of structured and unstructured data, cloud data services, and mobile and virtual worlds. We also report on discussions of the community’s growth, including suggestions for changes in community processes to move the research agenda forward, and to enhance impact on a broader audience.

1. A Turning Point in Database Research
Over the last twenty years, small groups of database researchers have periodically gathered to assess the state of the field and propose directions for future research [BDD+89, SSU91, ASU95, AZ+96, BBC+98, AAB03]. Reports of these meetings were written to serve various functions: to foster debate within the database research community, to explain research directions to external organizations, and to help focus community efforts on timely challenges.

This year, the tenor of the meeting was unusual and quite clear: database research and the data management industry are at a turning point, with unusually rich opportunities for technical advances, intellectual achievement, entrepreneurship and impact on science and society. Given the large number of opportunities, it is important for the research community to address issues that maximize impact within the field, across computing, and in external fields as well.

The sense of change in the air emerged quickly in the meeting, as a function of several factors:

1. **Breadth of excitement about Big Data.** In recent years, the number of communities working with large volumes of data has grown considerably, to include not only traditional enterprise applications and Web search, but also “e-science” efforts (in astronomy, biology, earth science, etc.), digital entertainment, natural language processing, social network analysis, and more. While the user base for traditional Database Management Systems (DBMSs) is growing quickly, there is also a groundswell of efforts to design new custom data management solutions from simpler components. The ubiquity of Big Data is significantly expanding the base of both users and developers of data management technologies, and will undoubtedly shake up the field.

2. **Data analysis as a profit center:** In traditional enterprise settings, the barriers between the IT department and business units are quickly dropping, and there are many examples of companies where the data is the business. As a consequence, data capture, integration and analysis are no longer considered a business cost; they are the keys to efficiency and profit. The industry supporting data analytics is growing quickly as a result. Corporate acquisitions of Business Intelligence (BI) vendors
alone last year totaled over 10 billion dollars, and that is only the “front end” of the data analytics toolchain. The market pressures for better analytics also bring new users and demands to the technology. Statistically sophisticated analysts are being hired in a growing number of industries, and are increasingly interested in running their formulae on the raw data. At the same time, a growing number of non-technical decision-makers want to “get their hands on the numbers” as well.

3. **Ubiquity of structured and unstructured data.** There is an explosion of structured data available on the Web and on enterprise intranets. This data comes from a variety of sources beyond traditional databases: large-scale efforts to extract structured information from text, software logs and sensors, and crawls of Deep Web sites. There is also an explosion of text-focused semi-structured data in the public domain in the form of blogs, Web 2.0 communities and instant messaging. And new incentive structures and web sites have emerged for publishing and curating structured data in a shared fashion as well. Current text-centric approaches to managing this data are easy to use, but ignore latent structure in the data that can add significant value. The race is on to develop techniques that can extract useful data from mostly noisy text and structured corpora, enable deeper explorations into individual datasets, and connect datasets together to wring out as much value as possible.

4. **Expanded developer demands.** Programmer adoption of relational DBMSs and query languages has grown significantly in recent years. This has been accelerated by the maturation of open source systems like MySQL and PostgreSQL, and the growing popularity of object-relational mapping packages like Ruby on Rails. However, the expanded user base brings new expectations for programmability and usability from a larger, broader and less specialized community of programmers. Some of these developers are unhappy or unwilling to “drop into” SQL, and view DBMSs as heavyweight to learn and manage relative to other open source components. As the ecosystem for database management evolves further beyond the typical DBMS user base, opportunities emerge for new programming models and for new system components for data management and manipulation.

5. **Architectural shifts in computing.** At the same time that user scenarios are expanding, computing substrates for data management are shifting rapidly. At the macro scale, the rise of “cloud” computing services suggests fundamental changes in software architecture. It democratizes access to parallel clusters of computers: every programmer now has the opportunity and motivation to design systems and services that can scale out incrementally to arbitrary degrees of parallelism. At a micro scale, computer architectures have shifted the focus of Moore’s Law from increasing clock speed per chip to increasing the number of processor cores and threads per chip. In storage technologies, major changes are underway in the memory hierarchy, due to the availability of more and larger on-chip caches, large inexpensive RAM, and flash memory. Power consumption has become an increasingly important aspect of the price/performance metric of large systems. These hardware trends alone motivate a wholesale reconsideration of data management software architecture.

Taken together, these factors signal an urgent, widespread need for new data management technologies. The opportunity for impact is enormous.

Traditionally, the database research community is known for impact: relational databases are emblematic of technology transfer. But in recent years, our externally visible impact has not evolved sufficiently beyond traditional database systems and enterprise data management, despite the expansion of our research portfolio. In the current climate, the community must recommit itself to impact and breadth. Impact is evaluated by external measures, so success will involve helping new classes of users, powering new computing platforms, and making conceptual breakthroughs across computing. These should be the motivating goals for the next round of database research.

To achieve these goals, two promising approaches that came up in discussion are what we call **reformation** and **synthesis**. The reformation agenda involves deconstructing core data-centric ideas and systems, and re-forming them for new applications and architectural realities. Part of this entails focusing outside the traditional RDBMS stack and its existing interfaces, emphasizing new data management systems for growth areas like e-science. In addition, database researchers should take data-centric ideas (declarative programming, query optimization) outside their original context in storage and retrieval, and attack new areas of computing where a data-centric mindset can have major impact. The synthesis agenda is intended to leverage good research ideas in areas that have yet to develop identifiable, agreed-upon system architectures, e.g., data integration, information extraction, data privacy, etc. The time is ripe for various
sub-communities to move out of the conceptual and algorithmic phase, and work together on comprehensive artifacts (systems, languages, services) that combine multiple techniques to solve complex user problems. Efforts toward synthesis can serve as rallying points for the research, will likely lead to new challenges and breakthroughs, and can increase the overall visibility of the work.

2. Research Opportunities

After two days of intense discussion, it was surprisingly easy for the group to reach consensus on a set of research topics to highlight for investigation in coming years. This is indicative of unusually exciting times.

Before presenting those topics, we stress a few points regarding what is not on this list. First, while we tried to focus on new opportunities, we do not propose they be pursued at the expense of existing good work. A number of areas we deemed critical were left out of this list because they have already become focus topics in the community. Many of these were mentioned in a previous report of this sort (see the Appendix), and/or are the subject of significant efforts in recent years. These ongoing efforts require continued investigation and funding. Second, we chose to keep the list short, favoring focus over coverage. Most of the authors have other promising research topics they would have liked to discuss at greater length here, but we chose to focus on topics that attracted the broadest interest in the group.

In addition to the list below, the main issues and areas that were raised repeatedly during the meeting include management of uncertain information, data privacy and security, e-science and other scholarly applications, human-centric interactions with data, social networks and Web 2.0, personalization and contextualization of query- and search-related tasks, streaming and networked data, self-tuning and adaptive systems, and the challenges raised by new hardware technologies and energy constraints. Most of these issues are in fact captured in some aspect of the discussion below, and many of them cut across multiple highlighted topics.

2.1. Revisiting Database Engines

System R and Ingres pioneered the architecture and algorithms of relational databases, and current commercial databases are still based on their designs. But the many changes in applications and technology described in Section 1 demand a reformation of the entire system stack for data management. Current big-market relational database systems have well-known limitations. While they provide a broad range of features, they have very narrow regimes where they provide peak performance: OLTP systems are tuned for lots of small, concurrent transactional debit/credit workloads, while decision-support systems are tuned for few read-mostly, large join and aggregation workloads. Meanwhile, there are many popular data-intensive tasks from the last decade for which relational databases provide poor price/performance and have been rejected: critical scenarios include text indexing, serving web pages, and media delivery. New workloads are emerging in sciences and Web 2.0-style applications, among other environments, where database engine technology could prove useful, but not as bundled in current database systems.

Even within traditional application domains, the current marketplace suggests that there is room for significant innovation. In the analytics markets for business and science, customers can buy petabytes of storage and thousands of processors, but the dominant commercial database systems cannot scale that far for many workloads. Even when they can, the cost of software and management relative to hardware is exorbitant. In the on-line transaction processing (OLTP) market, business imperatives like regulatory compliance and rapid response to changing business conditions raise the need to address data lifecycle issues such as data provenance, schema evolution, and versioning.

Given all these requirements, the commercial database market is wide open to new ideas and systems, and this is reflected in the funding climate for entrepreneurs. It is hard to remember a time when there were so many database engine startup companies. The market will undoubtedly consolidate over time, but things are changing fast right now, and it is a good time to try radical ideas.
Some research projects have begun taking revolutionary steps in database system architecture. There have been two distinct directions: broadening the useful range of applicability for multi-purpose database systems (e.g., to incorporate streams, text search, XML, information integration), and radically improving performance by designing special-purpose database systems for specific domains (e.g., streams, read-mostly analytics, and XML). Both directions have merit, and the evident commonality of focus suggests that these efforts may be synergistic: special-purpose techniques (e.g., new storage/compression formats) may be reusable in more general-purpose systems, and general-purpose architectural components or harnesses (e.g., extensible query optimizer frameworks) may enable new special-purpose systems to be prototyped more quickly.

Important research topics in the core database engine area include: (a) designing systems for clusters of many-core processors, which will exhibit limited and non-uniform access to off-chip memory; (b) exploiting remote RAM and Flash as persistent media, rather than relying solely on magnetic disk; (c) treating query optimization and physical data layout as a unified, adaptive, self-tuning task to be carried out continuously; (d) compressing and encrypting data at the storage layer, integrated with data layout and query optimization; (e) designing systems that embrace non-relational data models, rather than “shoehorning” them into tables; (f) trading off consistency and availability for better performance and scaleout to thousands of machines; (g) designing power-aware DBMSs that limit energy costs without sacrificing scalability.

That list of topics is not exhaustive. One industrial participant noted that this is a time of particular opportunity for academic researchers: the landscape has shifted enough that access to industrial legacy code provides little advantage, and large-scale clustered hardware is now rentable in “the cloud” at low cost. Moreover, industrial players and investors are actively looking for bold new ideas. This opportunity for academics to lead in system design is a major change in the research environment.

2.2. Declarative Programming for Emerging Platforms

Programmer productivity is a key challenge in computing. This has been acknowledged for many years, with the most notable mention in the database context being in Jim Gray’s Turing lecture of ten years ago. Today, the urgency of the problem is literally increasing exponentially as programmers target ever more complex environments, including manycore chips, distributed services, and cloud computing platforms. Non-expert programmers need to be able to easily write robust code that scales out across processors in both loosely- and tightly-coupled architectures.

Although developing new programming paradigms is not a database problem per se, ideas of data independence, declarative programming and cost-based optimization provide a promising angle of attack. There is significant evidence that data-centric approaches can have major impact on programming in the near term.

The recent popularity of Map-Reduce is one example of this potential. Map-Reduce is attractively simple, and builds on language and data-parallelism techniques that have been known for decades. For database researchers, the significance of Map-Reduce is in demonstrating the benefits of data-parallel programming to new classes of developers. This opens opportunities for our community to extend its impact, by developing more powerful and efficient languages and runtime mechanisms that help these developers address more complex problems.

As another example, new declarative languages, often grounded in Datalog, have recently been developed for a variety of domain-specific systems, in fields as diverse as networking and distributed systems, computer games, machine learning and robotics, compilers, security protocols, and information extraction. In many of these scenarios, the use of a declarative language reduced code size by orders of magnitude, while also enabling distributed or parallel execution. Surprisingly, the groups behind these various efforts have coordinated very little – the move to revive declarative languages in these new contexts has grown up organically.
A third example arises in enterprise application programming. Recent language extensions like Ruby on Rails and LINQ encourage query-like logic in programmer design patterns. But these packages have yet to seriously address the challenge of programming across multiple machines. For enterprise applications, a key distributed design decision is the partitioning of logic and data across multiple “tiers”: web clients, web servers, application servers, and a backend DBMS. Data independence is particularly valuable here, to allow programs to be specified without making a priori, permanent decisions about physical deployment across tiers. Automatic optimization processes could make these decisions, and move data and code as needed to achieve efficiency and correctness. XQuery has been proposed as one existing language that can facilitate this kind of declarative programming, in part because XML is often used in cross-tier protocols.

It is unusual to see this much energy surrounding new data-centric programming techniques, but the opportunity brings challenges as well. Among the research questions we face are language design, efficient compilers and runtimes, and techniques to optimize code automatically across both the horizontal distribution of parallel processors, and the vertical distribution of tiers. It seems natural that the techniques behind parallel and distributed databases – partitioned dataflow, cost-based query optimization – should extend to new environments. However, to succeed, these languages will have to be fairly expressive, going beyond simple Map-Reduce and Select-Project-Join-Aggregate dataflows. There is a need for “synthesis” work here to harvest useful techniques from the literature on database and logic programming languages and optimization, and to realize and extend them in new programming environments.

To have impact, our techniques also need to pay attention to the softer issues that capture the hearts and minds of programmers, such as attractive syntax, typing and modularity, development tools, and smooth interactions with the rest of the computing ecosystem (networks, files, user interfaces, web services, other languages, etc.)

Attacking this agenda requires database research to look outside its traditional boundaries and find allies across computing. It is a unique opportunity for a fundamental “reformation” of the notion of data management: not as a storage service, but as a broadly applicable programming paradigm.

2.3. The Interplay of Structured and Unstructured Data

A growing number of data management scenarios involve both structured and unstructured data. Within enterprises, we see large heterogeneous collections of structured data linked with unstructured data such as document and email repositories. On the World-Wide Web, we are witnessing a growing amount of structured data coming primarily from three sources: (1) millions of databases hidden behind forms (the deep web), (2) hundreds of millions of high-quality data items in HTML tables on web pages, and a growing number of mashups providing dynamic views on structured data, and (3) data contributed by Web 2.0 services, such as photo and video sites, collaborative annotation services and online structured-data repositories.

A significant long-term goal for our community is to transition from managing traditional databases consisting of well-defined schemata for structured business data, to the much more challenging task of managing a rich collection of structured, semi-structured and unstructured data, spread over many repositories in the enterprise and on the Web. This has sometimes been referred to as the challenge of managing dataspaces.

On the Web, our community has contributed primarily in two ways. First, we developed technology that enables the generation of domain-specific (“vertical”) search engines with relatively little effort. Second, we developed domain-independent technology for crawling through forms (i.e., automatically submitting well-formed queries to forms) and surfacing the resulting HTML pages in a search-engine index. Within the enterprise, we have recently made contributions to enterprise search and the discovery of relationships between structured and unstructured data.

The first challenge we face is to extract structure and meaning from unstructured and semi-structured data. Information Extraction technology can now pull structured entities and relationships out of unstructured text, even in unsupervised web-scale contexts. We expect hundreds of extractors being applied to a given
data source. Hence we need techniques for applying and managing predictions from large numbers of independently developed extractors. We also need algorithms that can introspect about the correctness of extractions and therefore combine multiple pieces of extraction evidence in a principled fashion. We are not alone at these efforts; to contribute in this area, the community should continue to strengthen its ties with the Information Retrieval and Machine Learning communities.

A significant aspect of the semantics of the data is its context. The context can have multiple forms, such as the text and hyperlinks that surround a table on a web page, the name of the directory in which data is stored and accompanying annotations or discussions, and relationships to physically or temporally proximate data items. Context helps interpret the meaning of data in such applications because the data is often less precise than in traditional database applications since it is extracted from unstructured text, is extremely heterogeneous, or is sensitive to the conditions under which it was captured. Better database technology is needed to manage data in context. In particular, we need techniques to discover data sources, to enhance the data by discovering implicit relationships, to determine the weight of an object’s context when assigning it semantics, and to maintain the provenance of data through these various steps of storage and computation.

The second challenge is to develop methods for effectively querying and deriving insight from the resulting sea of heterogeneous data. A specific problem is to answer keyword queries over large collections of heterogeneous data sources. We need to analyze the query to extract its intended semantics, and route the query to the relevant source(s) in the collection. Of course, keyword queries are just one entry point into data exploration, and there is a need for techniques that lead users into the most appropriate querying mechanism. Unlike previous work on information integration, the challenges here are that we do not assume we have semantic mappings for the data sources and we cannot assume that the domain of the query or the data sources is known. We need to develop algorithms for providing best-effort services on loosely integrated data. The system should provide some meaningful answers to queries with no need for any manual integration, and improve over time in a “pay-as-you-go” fashion as semantic relationships are discovered and refined. Developing index structures to support querying hybrid data is also a significant challenge. More generally, we need to develop new notions of correctness and consistency in order to provide metrics and to enable users or system designers to make cost/quality tradeoffs. We also need to develop the appropriate systems concepts around which to tie these functionalities.

In addition to managing existing data collections, we also have an opportunity to innovate on creating data collections. The emergence of Web 2.0 creates the potential for new kinds of data management scenarios in which users join ad-hoc communities to create, collaborate, curate and discuss data online. Since such communities will rarely agree on schemata ahead of time, they will need to be inferred from the data and will be highly dynamic; however they will still be used to guide users to consensus. Systems in this context need to incorporate visualizations effectively, because visualizations drive the exploration and analysis. Most importantly, these systems need to be extremely easy to use. This will probably require compromising on some typical database functionality and providing more semi-automatic “hints” that are mined from the data. There is an important opportunity for a feedback loop here – as more data gets created with such tools, information extraction and querying could become easier. Commercial and academic prototypes are beginning to appear in this arena, but there is plenty of space for additional innovation and contributions.

2.4. Cloud Data Services
Economic factors are leading to the rise of infrastructures providing software and computing facilities as a service, typically known as cloud services or cloud computing. Cloud services can provide efficiencies for application providers, both by limiting up-front capital expenses, and by reducing the cost of ownership over time. Such services are typically hosted in a data center, using shared commodity hardware for computation and storage. There is a varied set of cloud services available today, including application services (salesforce.com), storage services (Amazon S3), compute services (Google App Engine, Amazon EC2) and data services (Amazon SimpleDB, Microsoft SQL Server Data Services, Google’s Datstore). These services represent a variety of reformations of data management architectures, and more are on the horizon. We anticipate that many future data-centric applications will leverage data services in the cloud.
A cross-cutting theme in cloud services is the trade-off that providers face between functionality and operational costs. Today’s early cloud data services offer an API that is much more restricted than that of traditional database systems, with a minimalist query language and limited consistency guarantees. This pushes more programming burden on developers, but allows cloud providers to build more predictable services, and offer service level agreements that would be hard to provide for a full-function SQL data service. More work and experience will be needed on several fronts to explore the continuum between today’s early cloud data services and more full-functioned but possibly less predictable alternatives.

Manageability is particularly important in cloud environments. Relative to traditional systems, it is complicated by three factors: limited human intervention, high-variance workloads, and a variety of shared infrastructures. In the majority of cases, there will be no DBAs or system administrators to assist developers with their cloud-based applications; the platform will have to do much of that work automatically. Mixed workloads have always been difficult to tune, but may be unavoidable in this context. Even a single customer’s workload can vary widely over time: the elastic provisioning of cloud services makes it economical for a user to occasionally harness orders of magnitude more resources than usual for short bursts of work. Meanwhile, service tuning depends heavily upon the way that the shared infrastructure is “virtualized”. For example, Amazon EC2 uses hardware-level virtual machines as the programming interface. On the opposite end of the spectrum, salesforce.com implements “multi-tenant” hosting of many independent schemas in a single managed DBMS. Many other virtualization solutions are possible. Each has different visibility into the workloads above and platforms beneath, and different abilities to control each. These variations will require revisiting traditional roles and responsibilities for resource management across layers.

The need for manageability adds urgency to the development of self-managing database technologies explored in the last decade. Adaptive, online techniques will be required to make these systems viable, while new architectures and APIs – including the flexibility to depart from traditional SQL and transactional semantics when prudent – may motivate increasingly disruptive approaches to adaptivity.

The sheer scale of cloud computing presents its own challenges. Today’s SQL databases simply cannot scale to the thousands of nodes being deployed in the cloud context. On the storage front, it is unclear whether to address these limitations with different transactional implementation techniques, different storage semantics, or both. The database literature is rich in proposals on these issues. Current cloud services have begun to explore some simple pragmatic approaches, but more work is needed to synthesize ideas from the literature in modern cloud computing regimes. In terms of query processing and optimization, it will not be feasible to exhaustively search a plan space that considers thousands of processing sites, so some limitations on either the plan space or the search will be required. Finally, it is unclear how programmers will express their programs in the cloud, as mentioned in Section 2.2. More work is needed to understand the scaling realities of cloud computing – both performance constraints and application requirements – to help navigate this design space.

The sharing of physical resources in a cloud infrastructure puts a premium on data security and privacy, which cannot be guaranteed by physical boundaries of machines or networks. Hence cloud services provide fertile ground for efforts to synthesize and accelerate the work our community has done in these domains. The key to success in this arena will be to specifically target usage scenarios in the cloud, seated in practical economic incentives for service providers and customers.

As cloud data services become popular, we expect that new scenarios will emerge with their own challenges. For example, we anticipate the appearance of specialized services that are pre-loaded with large data sets, e.g., stock prices, weather history, web crawls, etc. The ability to “mash up” interesting data from private and public domains will be increasingly attractive, and will provide further motivation for the challenges in Section 2.3. This also points to the inevitability of services reaching out across clouds. This issue is already prevalent in scientific data “grids”, which typically have large shared data servers at multiple different sites, even within a single discipline. It also echoes, in the large, the standard proliferation of data sources in most enterprises. Federated cloud architectures will only enhance the challenges described above.
2.5. Mobile Applications and Virtual Worlds

There is a new class of applications, exemplified by mobile services and virtual worlds, characterized by the need to manage massive amounts of diverse user-created data, synthesize it intelligently, and provide real-time services. The data management community is beginning to understand the challenges these applications face, but much more work is needed. Accordingly, the discussion about these topics at the meeting was more speculative than about those of the previous sections, but we felt they deserve attention.

In the mobile space, we are witnessing two important trends. First, the platforms on which to build mobile applications (i.e., the hardware, software and network) are maturing to the point that they have attracted large user bases, and can ubiquitously support very powerful interactions “on the go”. Second, the emergence of mobile search and social networks suggests an exciting new set of mobile applications. These applications will deliver timely information (and advertisements) to mobile users depending on their location, personal preferences, social circles and extraneous factors (e.g., weather), and in general the context in which they operate. Providing these services requires synthesizing user input and behavior from multiple sources to determine user location and intent.

Virtual worlds like Second Life are growing quickly in popularity, and in many ways mirror the themes of mobile applications. While they began as interactive simulations for multiple users, they increasingly blur the distinctions with the real world, and suggest the potential for a more data-rich mixture. The term co-space is sometimes used to refer to a co-existing space for both virtual and physical worlds. In a co-space, locations and events in the physical world are captured by a large number of sensors and mobile devices, and materialized within a virtual world. Correspondingly, certain actions or events within the virtual world can have effects in the physical world (e.g., shopping or product promotion and experiential computer gaming). Applications of co-space include rich social networking, massive multi-player games, military training, edutainment and knowledge sharing.

In both of these areas, large amounts of data are flowing from users, being synthesized and used to affect the virtual and/or real world. These applications raise new challenges, such as a need to process heterogeneous data streams in order to materialize real-world events, the need to balance privacy against the collective benefit of sharing personal real-time information, and the need for more intelligent processing to send interesting events in the co-space to someone in the physical world. The programming of virtual actors in games and virtual worlds requires large-scale parallel programming, and declarative methods have been proposed as a solution in that environment as discussed in Section 2.2. These applications also require the development of efficient systems as suggested in Section 2.1, including appropriate storage and retrieval methods, data processing engines, parallel and distributed architectures, and power-sensitive software techniques for managing the events and communications across huge number of concurrent users.

3. Moving Forward

In addition to research topics, the meeting involved discussions of the research community’s processes, including the organization of publication procedures, research agendas, attraction and mentorship of new talent, and efforts to ensure research impact.

Prior to these discussions, a bit of ad hoc data analysis was performed over database conference bibliographies from the DBLP repository. While the effort was not scientific, the results indicated that the database research community has doubled in size over the last decade. Various metrics suggested this: the number of published papers, the number of distinct authors, the number of distinct institutions to which these authors belong, and the number of session topics at conferences, loosely defined. This served as a backdrop to the discussion that followed.

The community growth is placing pressure on research publications. At a topical level, the increasing technical scope of the community makes it difficult to keep track of the field. As a result, survey articles and tutorials are becoming an increasingly important contribution to the community. They should be encouraged both informally within the community, and via professional incentive structures such as tenure.
and promotion. In terms of processes, the reviewing load for papers is growing increasingly burdensome, and there was a perception that the quality of reviews had been decreasing over time. It was suggested that the lack of face-to-face PC meetings in recent years has exacerbated the problem of poor reviews, and removed opportunities for risky or speculative papers to be championed effectively over well-executed but more pedestrian work. Recent efforts to enhance the professionalism of papers and the reviewing process were discussed in this context. Many participants were skeptical that these efforts have had a positive effect on long-term research quality, as measured in intellectual and practical impact. At the same time, it was acknowledged that the community’s growth increases the need for clear and clearly-enforced academic processes. The challenge going forward is to find policies that simultaneously reward big ideas and risk-taking, while providing clear and fair rules for achieving those rewards. The publication venues would do well to focus on the first of those goals as much as they have focused recently on the second.

In addition to tuning the mainstream publication venues, there is opportunity to take advantage of other channels of communication. The database research community has had little presence in the relatively active market for technical books. Given the growing population of developers working with big data sets, there is a need for approachable books on scalable data management algorithms and techniques that programmers can use to build their own software. The current crop of college textbooks is not targeted at that market. There is also an opportunity to present database research contributions as big ideas in their own right, targeted at intellectually curious readers outside the specialty. In addition to books, electronic media like blogs and wikis can complement technical papers, by opening up different stages of the research lifecycle to discussion: status reports on ongoing projects, concise presentation of big ideas, vision statements and speculation. Online fora can also spur debate and discussion, if made appropriately provocative. Electronic media underscore the modern reality that it is easy to be widely published, but much harder to be widely read. This point should be remembered in the mainstream publication context as well, both by authors and reviewers. In the end, the consumers of an idea define its impact.

Given the growth in the database research community, the time is ripe for ambitious community-wide projects to stimulate collaboration and cross-fertilization of ideas. One proposal is to foster more data-driven research by building a globally shared collection of structured data, accepting contributions from all parties. Unlike previous efforts in this vein, the collection should not be designed for any particular benchmark – in fact, it is likely that most of the interesting problems suggested by this data are yet to be identified. There was also discussion of the role of open source software development in the database community. Despite a tradition of open-source software, academic database researchers at different institutions have relatively rarely reused or shared software. Given the current climate, it might be useful to move more aggressively toward sharing software, and collaborating on software projects across institutions. Information integration was mentioned as an area in which such an effort is emerging. Finally, interest was expressed in technical competitions akin to the Netflix challenge and KDD Cup competitions. To kick this effort off in the database domain, two areas were identified as ripe for competitions: system components for cloud computing (likely measured in terms of efficiency), and large-scale information extraction (likely measured in terms of accuracy and efficiency). While it was noted that each of these proposals requires a great deal of time and care to realize, several participants at the meeting volunteered to initiate efforts in these various directions. That work has begun, and participation from the broader community will be needed to help it succeed.

References


Appendix: Topics From Past Self-Assessments

Meetings to assess the state of database research were held in 1988 [BD+89], 1990 [SSU91], 1995 [ASU96], 1996 [AZ+], 1998 [BBC+98], and 2003 [AAB+03]. Each report describes changes in the application and technology landscape that motivate the need for new research. We summarize the driving forces in Table 1.

<table>
<thead>
<tr>
<th>Year</th>
<th>Driving Forces</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>Future Applications: CASE, CIM, images, spatial, information retrieval</td>
</tr>
<tr>
<td>1990</td>
<td>Future Applications: NASA data, CAD, genetics, data mining, multimedia</td>
</tr>
<tr>
<td>1995</td>
<td>Future Applications: NASA data, e-commerce, health care, digital publishing, collaborative design Technology Trends: hardware advances, database architecture changes (client-server, object-relational), the Web</td>
</tr>
<tr>
<td>1996</td>
<td>Future Applications: instant virtual enterprise, personal information systems</td>
</tr>
<tr>
<td>1998</td>
<td>Technology Trends: the Web, unifying program logic and database systems, hardware advances (scale up to megaservers, scale down to appliances)</td>
</tr>
<tr>
<td>2003</td>
<td>Future Applications: cross-enterprise applications, the sciences Technology Trends: hardware advances, maturation of related technologies (data mining, information retrieval)</td>
</tr>
</tbody>
</table>

Each report then goes on to enumerate particular research problems that need more investigation. Not surprisingly, many database research problems reappear in multiple reports. Usually, each occurrence is in the context of a different application scenario. For example, information integration has been recommended in the context of heterogeneous distributed databases (1990), better information distribution (1995), web-scale database integration (1998) and on-the-fly fusion of sensor data (2003). Although the topic recurs, the technical goals in each scenario usually differ. In Table 2, we summarize these recurring topics.

In many cases, these topics later became major database research fields. Examples include data mining, multimedia, integrating information retrieval and databases, data provenance, sensors and streaming, and probabilistic databases. It is impossible to know the extent to which these reports were a factor in these developments.

Some reports were more outwardly focused to non-database researchers. These reports summarized the field’s major accomplishments and pointed to worthwhile on-going research topics. We did not include
them in Table 1, which focuses only on areas that were felt to be under-researched at the time of the assessment report.

Necessarily, we applied a fair bit of editorial judgment in grouping topics. There were some topics that were recommended in one report but did not naturally group with topics in other reports. They are listed here for completeness: logical DB design tools, accounting and billing, site autonomy, operating system support for databases, personalization, and scientific data management.

Table 2 Recurring Topics in Database Research Assessment Meetings

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Version &amp; configuration management, repositories</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>More data types: Image, spatial, time, genetics, …</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information retrieval</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extendible DBMSs, object-oriented DBMSs</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exploit hardware advances</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query optimization</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallelism, scale-up, scale-out</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automated database administration</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High availability, replication</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workflow models, long transactions, workflow engines</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Active databases, rules, scalable trigger system</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heterogeneous DBMSs, interoperation, semantic consistency, data fusion, data provenance, data warehouses, mediators, info discovery, information exchange</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Uncertain and probabilistic data, data quality, query processing as evidence accumulation</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Schema-less DBs, integrating structured &amp; semi-structured data, DBMS architecture to integrate text, data, code and streams</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security and privacy, trustworthiness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data mining</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easier application development, visual programming tools, programming language interfaces, component models</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tertiary storage, 100 year storage</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real-time DBs, streams, sensor networks</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multimedia: quality of service, queries, UI support</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User interfaces for DBs</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The future of database technology is in the clouds
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVERVIEW</td>
<td>1</td>
</tr>
<tr>
<td>CLOUD COMPUTING ARRIVES</td>
<td>1</td>
</tr>
<tr>
<td>THE FUTURE OF ON-PREMISES DATABASE SYSTEMS: BIG CHALLENGES</td>
<td>1</td>
</tr>
<tr>
<td>THE FUTURE OF DATABASES: CLOUD DATABASE SERVICES</td>
<td>1</td>
</tr>
<tr>
<td>INTRODUCING DATABASE.COM</td>
<td>2</td>
</tr>
<tr>
<td>PROVEN</td>
<td>2</td>
</tr>
<tr>
<td>EASY TO USE</td>
<td>2</td>
</tr>
<tr>
<td>TRUSTWORTHY</td>
<td>2</td>
</tr>
<tr>
<td>MODERN</td>
<td>2</td>
</tr>
<tr>
<td>OPEN</td>
<td>3</td>
</tr>
<tr>
<td>POWERFUL</td>
<td>3</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>3</td>
</tr>
</tbody>
</table>
Overview
Database technology is the persistence layer at the heart of all data-centric applications, the tier of software that’s in charge of organizing, protecting, and managing shared database access reliably, securely, and efficiently. This paper explores what the future holds for database technology. We’ll learn the significant challenges facing on-premises database systems as cloud computing adoption becomes mainstream, and how the inevitable future of databases is cloud database services that are management-free, automatically tuned and optimized, and elastic to variable resource demands. Finally, we’ll take a quick look at the Database.com cloud database service and some of its differentiating features that uniquely position it to support social and mobile apps.

Cloud computing arrives
Since the millennium, cloud computing has forever changed the landscape of the IT world because it provides enterprise-grade computing resources that are affordable and instantly available. Clouds provide straightforward access to IT resources. You just access as much resource as you need, when you need it, and never have to deal with the complexities of managing all the underlying mechanisms that provide the resources. Life is suddenly a lot simpler and easier with cloud computing.

The future of on-premises database systems: big challenges
For years, enterprises have been using database management systems (DBMSs) in their data centers as the heart of mission-critical applications. But the classic on-premises database server as we know it today will not likely survive in the world of cloud computing. Why such a bold statement?

- **On-premises database systems are difficult and time-consuming to manage** – They’re notoriously complicated to configure and maintain, which doesn’t match well with the low-touch model of cloud computing.
- **On-premises database systems don’t scale easily** – They’re not designed to automatically scale their workload elastically in response to varying demands at a moment’s notice.
- **On-premises database systems aren’t multitenant** – They’re inherently single-tenant systems, which requires that each organization set up and manage multiple databases, sometimes for each application, and that’s a lot of work.

While traditional database vendors struggle to adapt existing technology or invent new technology to address these problems, a serious challenge to their market share dominance has emerged: cloud database offerings.

The future of databases: cloud database services
A cloud database service, or Database as a Service (DbaaS), provides easy access to scalable, on-demand database resources that data-centric apps can use. Cloud database services are the inevitable future of database technology because, when designed and delivered properly, they unlock the potential to solve the most pressing and daunting problems IT shops now face when deploying and managing on-premises relational databases.

- **Cloud databases are easy to use** because they don’t require users to perform any low-level management such as patching, backups, and configuration.
- **Cloud databases deliver transparent performance** because they’re automatically tuned and improved by the service provider daily.
- **Cloud databases are cost-efficient** because they don’t require any up-front, expensive investments like software licenses or hardware, and they transparently scale according to varying application workloads.
- **Cloud databases are instantly available** to whoever needs them with just a few mouse clicks.
Cloud databases are **reliable and secure** because they’re constantly monitored and administered by dedicated professionals whose only job is to manage them.

Database as a Service is not just a concept—many nascent cloud database solutions are available or appearing today that embody the attributes described previously. But can you trust such untested systems to reliably and securely manage your data?

**Introducing Database.com**

Database.com is a proven cloud database solution you can use to support the persistence needs for your apps. Following are some of the key features of Database.com.

**Proven**

Database.com is proven database technology that powers all of salesforce.com’s products today. At the time of this writing, it serves more than more than 100,000 organizations, 135,000 applications, 3 million users, and 10 billion transactions per quarter—all with an average request response time of less than 300ms and an average up time of 99.9+ percent for many years. Salesforce.com is completely transparent about the health of all systems via the website trust.salesforce.com. Here, you’ll find historical and current status and performance information.

**Easy to use**

With Database.com, there’s nothing to manage—salesforce.com takes care of everything for you. There’s no software to install, update, and patch. No waiting on someone else when you want to provision databases. No worries about database backup and disaster recovery. No complex documentation set with thousands of pages and parameters to tune for performance or elasticity. There’s even automatic indexing. Whether you have 1 database or 1,000 databases, all you need to focus on is building great apps.

**Trustworthy**

Database.com is built with the security and privacy of customer information in mind. Salesforce.com’s infrastructure and corporate workplace meet all of the highest industry standards, including SAS 70 Type II, SysTrust, and ISO 27001 certifications.

**Modern**

Database.com is more than just another database system—it’s jam-packed with next-generation features that make building and maintaining highly functional, secure, social, and mobile apps a snap.

- Database.com users, profiles, roles, groups, and row-level sharing rules help you build secure apps without the need to code, test, and maintain your own complicated security logic.

- With Database.com, it’s easy to implement common application functionality without writing complicated and error-prone code. Such features include declarative, point-and-click configuration for workflows, encrypted/masked fields, validation rules, formula fields, roll-up summary fields, and cross-object validation rules.

- Database.com is “social” because it includes the Salesforce Chatter API, a built-in data model apps can leverage to become instantly social and collaborative.

- Database.com’s REST APIs, OAuth implementation for user authentication/authorization, data feeds, custom Web services, embedded security model, and other features make it a perfect fit for easily building secure, scalable mobile apps, either native or HTML5.
Open
Database.com’s full complement of open APIs lets you build applications using the approach of your choice. REST- and SOAP-based APIs are standards-based APIs that make Database.com open to whatever programming language you want to use. Using various APIs, your applications can do many things such as create-read-update-delete (CRUD) business data, load a large number of records asynchronously, and take advantage of the Chatter API to provide collaboration and social networking capabilities to any application.

Powerful
Most modern apps use server-side logic to centralize complex business logic and enforce complex data integrity rules. Apex, with syntax much like Java, is Database.com’s procedural language that you can use to create server-side logic for an application. For example, Apex lets you create stored procedures that modify the database within the context of ACID transactions, and expose them as a custom Web services API (RESTful or SOAP) for your apps. You can also use Apex to build database triggers, routines that automatically fire (execute) when apps modify records in your database.

Summary
Since their inception in the 1970s, traditional on-premises database systems have grown overly complex, difficult to manage, and are struggling today to take full advantage of cloud computing technology. On the other hand, management-free, automatically tuned and optimized, and highly scalable cloud database services appear destined to be the future of databases.

Database.com is an enterprise cloud database service provided by salesforce.com. Database.com’s technology, designed specifically to service a cloud, today supports some 100,000+ organizations across the globe. Individual enterprises and commercial Web software vendors trust Database.com to deliver robust, reliable, secure, Internet-scale applications.