The Web in 2010: Challenges and Opportunities for Database Research

Gerhard Weikum

University of the Saarland, Saarbrücken, Germany weikum@cs.uni-sb.de, http://www-dbs.cs.uni-sb.de/

Abstract. The impressive advances in global networking and information technology provide great opportunities for all kinds of Web-based information services, ranging from digital libraries and information discovery to virtual-enterprise workflows and electronic commerce. However, many of these services still exhibit rather poor quality in terms of unacceptable performance during load peaks, frequent and long outages, and unsatisfactory search results. For the next decade, the overriding goal of database research should be to provide means for building zero-administration, self-tuning information services with predictable response time, virtually continuous availability, and, ultimately, "moneyback" service-quality guarantees. A particularly challenging aspect of this theme is the quality of search results in digital libraries, scientific data repositories, and on the Web. To aim for more intelligent search that can truly find needles in haystacks, classical information retrieval methods should be integrated with querying capabilities for structurally richer Web data, most notably XML data, and automatic classification methods based on standardized ontologies and statistical machine learning. This paper gives an overview of promising research directions along these lines.

1 Introduction

1.1 Blessings and Curses of the World Wide Web

We are witnessing the proliferation of the global information society with a sheer explosion of information services on the World Wide Web. This opens up unprecedented opportunities for information discovery, virtual enterprises and cyberspace-based collaboration, and also more mundane things such as e-commerce. Using such services is, however, often a frustrating experience. Many information services, including Web search engines, deliver poor results - inconsistent, arbitrarily inaccurate, or completely irrelevant data - to their clients, break easily and exhibit long outages, or perform so poorly that unacceptable response times ultimately render the offered service useless. The bottom line is that the quality of services is highly unpredictable, and service quality guarantees are usually absent in today's fast-moving IT world.

Contrast this situation with the evolution of database systems. Over the last three decades, database systems have developed an outstanding reputation for keeping huge amounts of mission-critical data consistent, virtually never losing data, providing high availability, excellent performance for a very large number of concurrently active clients under a wide variety of workload patterns, and so on. Indeed most mature e-commerce servers use a database system as a backend server, behind a middle-tier Web application server, for critical processing such as purchases. The most important assets that database technology provides to such Web services are [43]:

- 1. declarative data access through high-level query languages based on predicate logic and exemplified in the SQL standard and
- 2. automatic preservation of data consistency through atomic transactions in the presence of concurrent accesses and transient system failures.

These two fundamental contributions have been recognized by the computer science community with the Turing Awards to Ted Codd and Jim Gray. Database systems have proven that they are among the most dependable, intensively stress-tested services within computer science and the IT business. On the flip side of the coin, however, we have to admit that full-fledged database systems are typically heavy-weight platforms, and many of their salient qualities can be achieved only in conjunction with a human support staff that takes care of system administration and performance tuning [2,44].

Information retrieval technology, the database area's neighboring field, is a cornerstone of digital libraries and text-oriented and, to some extent, even multimedia-enhanced search engines for the World Wide Web and in Intranets. Most notably, Gerard Salton's vector space model is prevalent, in some form or another, in virtually all search engines [3,5,23]. So information retrieval is among the most useful Web technologies for the masses. But a closer look reveals that it heavily draws on heuristics and its effectiveness in satisfying the information needs of advanced users quite often faces its limits.

1.2 Towards a Science of Service Quality Guarantees

Database technology has the potential for being a backbone of quality-conscious information services. However, the rapidly evolving, highly diverse world of global information services calls for a new blend of database technology [44,2]. Often only certain components of a database system are needed as building blocks that have to be integrated with other technologies such as workflow, multimedia, or security technologies; a prominent example for an application class with this characteristic is e-commerce. In addition, with a database system's cost of ownership being dominated by human feed and care for administration and tuning, a critical prerequisite for ubiquitous penetration of database technology is that it be completely self-tuning, with automatic adaptation to evolving workload characteristics and "zero administration" in general [9,49].

The elusive goal that we should strive for would be a generalized notion of *guaranteed service quality*. The notion of service quality (also known as quality of

service or QoS for short) has come up in the context of multimedia communication (see, e.g., [22]), but has so far been limited to low-level issues like guaranteed packet delivery rates. What we need is a well-founded generalization of this notion to the application level, covering building blocks such as database access or middleware components like web application and workflow servers, Web or Intranet search engines, and also comprehensive, value-added information services on the Internet. These guarantees are given to the end-users of information services, but it must also be easy for service builders and providers to enforce the guarantees. More specifically, I envision the following three classes of guarantees:

- Response time of Web services must be predictable, and we must be able to guarantee that most service requests, say 99 percent, are served within user-acceptable time, say, 2 seconds, even during load peaks.
- As a result of globalization, services must be continuously available with downtimes limited to a few minutes per year. So services should be (more or less) always up.
- As a particularly important kind of Web service, search engines are becoming a key component for mastering the exploding volume and manifold of information in our modern society. Therefore, the quality of search results, in terms of completeness, accuracy, timeliness, and cost-effectiveness, needs to be drastically improved and should ideally be guaranteed (in a sense that will be discussed later).

Obviously, there are further aspects of service quality such as provably correct behavior or guaranteed security, but these will not be considered in this paper (as they are covered in other papers of this volume). Also note that quality guarantees are of fundamental importance in many areas of computer science (see, e.g., [19,50]), not just Web services, but I will focus on Web-related aspects in this paper.

1.3 Outline of the Paper

The rest of the paper is organized as follows. Section 2 discusses performance predictability and guarantees. Section 3 considers the goal of continuous availability. Section 4 discusses the quality of search results for Web search engines. All sections will first identify the state of the art, then outline the grand challenges, and finally point out promising research avenues.

2 Performance Guarantees

2.1 State of the Art

Today's lack of performance guarantees in Web services is often circumscribed by the nice term "best-effort performance", which essentially means that everything is unpredictable and nothing can be guaranteed. In general a number of components may cause delays: the client's connection to its Internet provider, the provider's proxy server, the wide-area network itself, the Web application server at the target site, or the content provider's backend data server. Usually, the network bandwidth is not the limiting bottleneck. Rather, the major reason for poor performance during peak hours is that requests suffer queueing delays at congested data servers or gateways. Mean response time may be acceptable when averaged over long time periods like a week or a month, and some application service providers even make contractual guarantees about such long-range metrics. However, these guarantees can be easily given yet do not tell much about the observed response time distribution during the most popular business hours, which is when performance matters most. The core of the problem lies in the Internet, application, or content providers' inability to predict the quantitative behavior of their servers at sufficient detail and to configure the server resources appropriately.

What we need is not just better response times in a general sense, but predictability and guarantees about the tail of the response time distribution. Ideally, we would wish to give an upper bound for the worst-case response time of client requests, but with the entire world being potential clients of a server, the workload can be characterized only statistically at best. Consequently the server resources needed for true worst-case guarantees would exceed all economically feasible limits. So response time guarantees should be stochastic, for example, specifying that the response time will be 2 seconds or less with probability 99 percent.

As an example for stochastic guarantees along these lines consider a media server that transfers video data streams to clients over an extended time period (i.e., the playback duration of a video). The server discretizes the continuous stream in a round-based manner, with round length T (e.g., one second), by periodically sending data fragments each of which holds enough data for say the next round of playback. For smooth playback at the client it is crucial that these fragments arrive just in time, to avoid glitches such as user-noticeable "hiccups". A complication is that video data is usually encoded with variable bit rate because of compression. So the size of fragments varies within a video (and also, of course, across video or audio objects). To ensure service quality for all simultaneously active streams the server employs an admission control. An additional data stream is admitted only if the fragment delivery deadlines can still be met for *all* subsequent fragments of *all* active streams including the new one.

For given server resources (e.g., number of disks, amount of memory) as well as data and workload profiles, the server needs to know the maximum number, N, of admissable concurrent streams. This limit can be derived in a conservative manner by assuming a maximum fragment size and worst-case disk access delays for all streams in all scheduling rounds (see, e.g., [22]). However, this crude assumption does not take into account the variability of the fragment size and the disks' performance behavior, and therefore tends to end up with a substantially underutilized server. A much better cost/performance ratio can be achieved by addressing this problem in a stochastic manner. What we need to compute is the maximum number of fragments, N, such that the total service time for fetching all N fragments from disk does not exceed the round length T with, specifiable, high probability (e.g., 99.99 percent). To this end, we decompose the service time into seek time, rotational delay, and transfer time - characterizing each in a stochastic manner, and then consider the convolution of these continuous random variables. Not surprisingly, this derivation is more tractable in terms of the Laplace transforms of the various distributions, and we can use results from large deviation theory, most notably, Chernoff bounds [35], to obtain an explicit result for the maximum sustainable number of streams [36]. The predictions and thus guarantees derived from this model are fairly accurate and can substantially improve the cost/performance of the server. In summary, such a stochastic model serves to configure the server, by choosing an appropriate limit for the admission control and determining the number of disks and the amount of memory such that the required service quality for the given workload can be guaranteed.

A similar methodology can be applied to predict the response time distribution of data requests to a conventional Web server that manages discrete data like text or image documents. An intriguing generalization that goes one step further is to consider mixed media servers that manage both continuous media and discrete data. For example, in teleteaching students should not simply watch tape-recorded lectures, but should also work with electronic textbooks, load demo programs, and so on. It is generally beneficial to hold both continuous and discrete data objects on a shared disk pool, so that load fluctuations can be smoothed to the best possible extent; otherwise, with two dedicated disk pools, we could end up with the discrete-data disks being overloaded at some point while at the same time the continuous-data disks could be underutilized. Such a load-sharing approach to a mixed media server requires special considerations on the disk scheduling of requests and a much more sophisticated stochastic model for predicting both the glitch rate of continuous data streams and the response time of discrete data accesses. A solution for this problem has been developed in [37].

2.2 Grand Challenge

Performance guarantees along the lines outlined above should by no means be reserved to multimedia applications, but should rather be an integrated part of all information services on the Web. Like in our examples, it is crucial that such guarantees capture the tail of the response time distribution, as opposed to referring merely to mean values. ¿From a customer viewpoint, these should be money-back guarantees: unacceptable performance results in no payment or compensation by the service provider. In 2010 all services that do not provide such guarantees should be out of business within their first month of operation.

I envision the following to become common practice. Vendors or application service hosts will give free hardware and software to content providers for some trial period (e.g., a few weeks). During this period the system will capture a profile of the workload and will determine an appropriate system configuration. Then a contract is set up that includes money-back performance guarantees between the vendor and the content provider. An analogous contract will be in effect between the content provider and end-user customers. The stochastic guarantee would translate into a test on a concrete sample of actual information requests by a customer: if the response time is unacceptable in more than say 5 percent of all requests then the customer will get her money back. As the workload may evolve over time, the various self-tuning servers should monitor the criticality of their stochastic promises, automatically analyze potential bottleneck, and alert the content or application service provider about necessary resource upgrades. Note that this way we no longer rely on human administrators that often realize only after a week of poor performance that some action is overdue.

2.3 Research Avenues

A new research community has recently been emerging under the name Web Engineering. Obviously, providing better performance is a top item on their agenda, studying issues like hierarchical and distributed caching and prefetching as well as optimizations to network protocols. However, as pointed out above, this is not far-reaching enough. Performance *guarantees* are needed, and they require predictability of the function that relates workload and server as well as network configuration to the resulting performance metrics. This topic is, of course, not at all new: performance assessment based on stochastic modeling has been a very strong field in the seventies, but it has now disappeared from most computer science curricula. I claim that stochastic models are the key to predicting and thus controlling Web service performance. Without such rigorous underpinnings Web Engineering will fail.

It is, of course, well known that the mathematics of stochastic model becomes extremely difficult and sometimes intractable when the models reach a certain complexity, especially when certain standard assumptions such as Poisson arrivals are no longer justified (see, e.g., [24]). In such situations, there are two options:

- We can attempt to analyze a conservative approximation of the real system that serves as an upper bound for the actual performance. For example, for the mixed media server sketched in Subsection 2.1 we were faced with a mix of periodic and Poisson arrivals and a fairly sophisticated, dynamic (i.e., state-dependent) scheduling policy. The actual analyis considered a static scheduling policy for tractability, and we carefully reasoned that the dynamic policy does never perform worse. In a similar vein, when a comprehensive mathematical solution is out of reach, we may resort to simulation results, with proper statistical confidence, for submodels and derive higherlevel metrics from such submodels using analytic methods.
- Sometimes the best remedy is to simplify the real system itself to make the analysis tractable. This would usually result in some performance losses, but the gain lies in predictability. With simpler building blocks absolute performance can often be boosted by simply scaling up hardware resources

(i.e., memory, disks, etc.), while still being able to give strong performance guarantees. For example, our mixed media server could be simplified by not exploiting any resource sharing between continuous and discrete data, essentially boiling down to separate servers for each of the two workload classes with disjoint resources. Then, of course, the analysis would become much simpler and even more accurate. This consideration can be viewed as an instance of the "pick the low hanging fruit" engineering rule of thumb. This is to say that often 90 percent of the solution (and the performance) can be achieved with 10 percent of the intellectual effort and resulting system complexity. Scientists should not be too proud to exploit this kind of pragmatic approach (notwithstanding the fact that long-term research towards the most general solution should be continued as well).

Unfortunately, too often neither of these options is pursued and rather the mathematical difficulty of the analysis is used as an excuse for merely providing "besteffort performance".

3 Continuous Availability

3.1 State of the Art

The availability of a server is the probability that a client request at an arbitrary timepoint will find the server listening and willing to process the request. The reason for temporarily being unavailable are transient failures and resulting downtimes. After a failure a server undergoes some recovery procedure, a "repair", and will then resume normal operation until the next failure occurs. In the long term (i.e., the stationary case with time approaching infinity) the availability of a server is given by the ratio $\frac{MTTF}{MTTF+MTTR}$ with MTTF denoting the mean time to failure and MTTR the mean time to repair.

To improve availability and approach the ideal value 1.0 the obvious approach seems to increase the MTTF. However, empirical observations tend to suggest that the MTTF cannot be advanced without limits. The main problem why servers occasionally crash are so-called Heisenbugs, virtually non-reproducible software errors that occur once in a while because of special race conditions or other synchronization bugs, exotic feature interactions, or inadequate configuration and administration settings in multi-threaded, heavily loaded servers under specific stress conditions [20]. Such bugs resemble Heisenberg's uncertainty relation: if we instrument the system for debugging then the bugs do not occur anymore. In many of the less mature Internet applications even Bohrbugs (the deterministic counterpart to Heisenbugs) are common because of poor debugging. Clearly, there is no real reason why it should not be possible to eliminate Heisenbugs (and Bohrbugs should definitely be eliminated by proper software engineering), but many existing information systems are so large and complex that even state-of-the-art verification and debugging methods will be inherently incomplete for quite a few years to come. In addition, Heisenbug problems are often aggravated by poor system administration, for example, inappropriate configuration or tuning settings. Note that periodic rebooting (e.g., once a week), which may be viewed as a prophylactic measure against Heisenbugs, leads to occasional downtime, too, and thus adversely affects availability.

The approaches that database systems have taken to cope with these problems are the following:

- Database servers should fail-stop upon the first indication of something going abnormal and provide fast and robust database recovery, based on transactional atomicity and persistence and highly optimized crash recovery algorithms [20,51].
- For mission-critical applications, data and server processes should be replicated on a backup site that can take over more or less immediately when the primary server fails. The backup may be within the same shared-disk cluster (i.e., in the same room or a neighboring building), or geographically remote (i.e., in a different city) if disaster recovery is an issue. In either case, the "failover" procedure by the backup involves rolling back active transactions and may thus be noticeable to application programs which then need special failure handling code.
- Whenever possible and affordable database systems should be administered by highly skilled and experienced (hence usually highly paid) humans who apply great care in system setup, monitoring, and occasional re-configuration.

These techniques have been developed to a fairly mature level in the database industry, and in conjunction with professional administration availability levels of 99.99 percent are the standard for mission-critical systems such as stock trading platforms or online banking. On first glance this figure seems impressive, but a closer look shows that it is still far from being satisfactory. 0.01 percent unavailability translates into approximately one hour downtime per year, and because of the Heisenbug phenomenon it is more likely that crashes hit during popular, heavy-load business hours. According to business analysts one minute downtime has a cost of up to a \$ 100,000 because of its impact on the affected company's market position [39]. Furthermore and most importantly, the 99.99 percent availability figure solely refers to the data server, which is only the backend in modern Web applications. Because application programs need to be execute their failure handling logic for retrying aborted transactions the actual outages observed by end-users are significantly longer (e.g., when application programs need to open new ODBC sessions or perform long computations within the transaction). Often such code is incomplete or missing, and then failures are exposed to the user, for example, by showing a message like "status code -23495: no such transaction" in the user's Internet browser. For an e-commerce service such behavior is fairly embarrassing (and more than inconvenient to the user) when this happens upon the final checkout with a full shopping cart. On the other hand, the application program or the user must not blindly re-initiate a transaction even if no positive return code has been received, as the transaction may nevertheless have succeeded and its effects are not necessarily idempotent. Users that are not sufficiently careful may end up buying the same book twice. So, in summary, user-transparent recovery is not well understood for entire e-services that comprise Internet connections and middle-tier application servers that communicate with one or more backend database servers and possibly also other application servers in a federated manner (e.g., to implement company-wide business portals or value-added broker services such as electronic travel agencies).

3.2 Grand Challenge

Given that downtime, at the user-perceivable level, is so expensive, bold steps towards continuous availability should have very high priority on our research agenda. The grand challenge for 2010 is to achieve less than one minute expected downtime per year, which is equivalent to 99.9999 percent availability, a two orders of magnitude improvement of unavailability and a good approximation of truly non-stop services.

The most futuristic aspect of this challenge lies in the fact that availability should be measured from the end-user's perspective. It is not good enough that the database server is up (again) but the application itself is not responding to the user. To this end, a comprehensive notion of recovery is needed that integrates data, process, and message recovery in an efficient manner. Recovery procedures at the various levels should be coordinated so that *all* failures can be masked to the human users. With complex, multi-tier and federated, system architectures for modern Web services, this challenge amounts to some kind of world-wide failure masking.

Finally, because failures will still occur but should be masked by replication and fast failover, availability must really be assessed in combination with performance. During failover procedures and while some primary servers are down, the backup servers have to process a higher load and thus exhibit a certain response time degradation. Systems should be configured such that response time guarantees can still be satisfied in the presence of such degraded phases.

3.3 Research Avenues

In principle, world-wide failure masking is "merely" a matter of logging all data updates, messages and other events, and periodically saving the state of all involved processes. Such approaches have been pursued in the fault tolerance community for a long time [11,20], but practical solutions have been limited to dedicated server complexes for special applications (e.g., stock trading) and typically have high overhead. The difficulty in scaling up these approaches lies in the subtleties of the interplay of many, largely autonomous, components in a multi-tier federation and their recovery behavior. Orchestrating a large number of highly distributed logging and recovery managers in an efficient manner with all failures masked will probably require new principles. The reply logging method in [33] may be a promising starting point, but has to be generalized from client-server systems to general, multi-tier, Web applications. Also, as recovery code is really critical and the devil is in the details, rigorous correctness reasoning for this code is mandatory. This may call for a new comprehensive theory of recovery.

In addition to these fundamental advances that I envision for the next decade, the overriding goal of reaching 99.9999 percent availability requires significant progress on the engineering side as well. In particular, error-prone system administration tasks need to be automated to the largest possible extent (as already pointed out, from a different perspective, in Section 2). Furthermore, software maintenance must be possible without interrupting system operation. For example, it should be possible to upgrade to a new version of the operating system without having to bring down the database system on the same computer.

Finally, the key to satisfying performance goals even when some replicated components are down and have failed over to backup components is to configure the overall system appropriately. Most importantly, the degrees of replication, for data and processes, determine not only the absolute availability but also the effective performance. Note that in advanced services that involve many components the probability that some (replica of a) component will be temporarily down is non-negligible, so that we cannot simply carry over the optimal performance (with all replicas of all components simultaneously running) to the real state(s) of the overall system. The necessary conditioning of the relevant performance metrics with the probabilities of the various system states leads to the notion of "performability", a concept that has been around in the performance modeling community for quite a while [24] but is otherwise virtually unknown. So we need stochastic models along these lines and configuration tools for performability guarantees. An initial approach in this direction, although limited to a specific system context, can be found in [18].

4 Search Result Quality

4.1 State of the Art

All search engines for the Web, Intranets, or digital libraries mostly build on the vector space model that views text documents (including HTML documents) as vectors of term relevance scores. These terms, also known as features, represent word occurrences in documents after stemming and other normalizations. The relevance score of a term in a document is usually derived from the term frequency (tf) in the document and the overall number of documents with this term or the corresponding total term frequency, the so-called inverse document frequency (idf), giving rise to (several variants of) the somewhat pragmatic, but empirically well proven tf*idf formula. Queries are vectors, too, and we can then use a similarity metric between vectors, for example, the Euclidean distance or the cosine metric, to produce a ranked list of search results, in descending order of relevance scores (i.e., estimations of what the user who posed the query would rate as relevant). The quality of a search results is assessed a posteriori in terms of the empirical metrics precision and recall: precision is the fraction of

11

truly relevant documents in the result or the top N matches in the result ranking (N typically being 10), and recall is the fraction of found documents out of the relevant documents that exist in the underlying corpus (or the entire Web). Experimentally studying precision and recall is itself a difficult issue.

A nice property of this classical approach is that it can be generalized to searching on multimedia objects such as images, videos, or music [14,22]. Once an appropriate feature space has been defined, for example, based on color distribution or contours in images, the principles of similarity ranking apply more or less directly. Of course, appropriate features heavily depend on the specific application. Current approaches on this issue still seem to be more of an art than scientific engineering. Notwithstanding this general assessment, there are some useful cases of multimedia similarity search in limited application contexts.

The above technology based on the vector space model is more than twenty years old and applicable to any kind of text document collection. But the Web is more than just a corpus of documents, given that its documents are extensively cross-related through hyperlinks, and also many Intranets are structured in such a fashion. A relatively recent trend has been to analyze the link structure between the documents, viewing the Web as a graph, and define the *authority* of Web sites or documents as an additional metric for search result ranking [5,28]. One way of doing this is to consider a random walk on (a large representative sample of) the Web where outgoing links are followed with uniform probabilities, adding up to $1-\epsilon$, and a random jump is performed with probability ϵ . Then the stationary probabilities of hitting a URL can be computed from the underlying discretetime Markov chain, and these probabilities are used as authority scores. There are alternative ways of defining and computing the notion of authority, but all lead to similar Eigenvalue problems. The point of this metric is that documents or sites with high authority should be preferred in search results, to achieve better precision (i.e., "to sort out the junk" in more colloquial terms).

It is important, however, to realize that authority is different from and complementary to the notion of relevance. In fact, search engines that make use of authority combine it with tf^{*}idf-based relevance scores, for example, by computing a weighted sum of authority and relevance scores where the weights are chosen in a pragmatic, more or less ad hoc, manner. Further note that authority assessment is of help mostly for popular queries that would otherwise produce a huge result list (e.g., searching for famous pop stars, actors, or politicians). It does not help much for advanced, expert-style queries for which it is difficult to find any useful results at all (i.e., for which recall is the problem). As an example, searching for "Chernoff theorem" on Web search engines that take into account authority metrics leads to hardly any useful results. Typically, these search engines return frequently cited documents such as conference home pages that do have some relationship to stochastics and sometimes even large deviation theory, but the actual information that one is looking for (i.e., the theorem that explains what Chernoff bounds are) can at best be reached by manually traversing outgoing links (e.g., textbooks referenced in papers that appear in the conference whose home page was found). Of course, the relevance estimation problem shows up in this example as well, and this holds for both search engines with and without authority assessment. For example, one search engine returns a document about Fermat's theorem among the top ten (obviously collapsing all theorems into one topic, which is probably considered exotic enough for the masses of Web users), another engine points to a "model" Mikki Chernoff, and even the better engines tend to deliver scientific publications that cite Chernoff but are otherwise only remotely related to the query itself. The most useful result, a textbook on large deviation theory, was found (among the top ten) only by Yahoo, which uses a manually/intellectually produced directory and is thus actually incomparable to the fully automated search engines. One can easily think of many more advance examples with similarly poor search results (e.g., searching for infrequent allergies, people who share an exotic hobby, descriptions of specific hikes off the beaten path, etc.)

4.2 Grand Challenge

The Web can be viewed as a huge, global knowledge base that potentially bears the answers to almost all information needs for almost everybody: common people who are interested in sports, traveling, or other hobbies, business people who are interested in market data and financial news, scientists who are interested in background material from neighboring fields and results related to their own work, and so on. Our ability to exploit this great potential and find high quality results for advanced queries are still fairly immature. In particular, advanced queries that are, so to speak, looking for needles in haystacks are poorly handled by today's search engines. These are queries that would find few results even under ideal conditions; so recall rather than precision is the main problem here.

For 2010 I hope to see tremendous progress on the capabilities for intelligent information search. We are indeed forced to tackle this as a grand challenge in science in general, for otherwise we will inevitably be swamped with information without being able to retrieve any useful results. This follows from a simple extrapolation of the rapid growth of information in the world and the observation that advances in information retrieval have been fairly incremental in the last two decades. Currently there are still ways of compensating poor search engine results by manually navigating in the extended neighborhood of some reasonably promising URL or resorting to intellectually maintained directories of the Yahoo style. However, these methods are very time-consuming, and become less and less affordable with intellectual time being the most precious resource. Even worse, there is no way for these manual methods to scale up with the rapid growth of the Web.

Examples of the kind of advanced queries that intelligent search engines should be able to effectively (and efficiently) handle in a decade would be:

 A hiker seeking for information on a particular cross-country (i.e., trail-less) route that is not available in any guidebooks but likely to be discussed on personal homepages of a few adventurous hikers.

- A programmer searching for publicly available code for a specific algorithm (e.g., a B⁺-tree with built-in concurrency control) in a specific language on a specific operating system.
- A mathematician (or computer scientist) who conjectures a certain result and searches for similar theorems that are already known (including more general theorems that cover the conjecture as a special case).
- A surgeon who prepares herself for a complicated brain surgery and searches for similar cases (including similar X-ray images or tomographies of tumors).

Ultimately, next-generation Web search engines should be able to find every piece of information that a human expert could retrieve if she had infinitely much time (and provided the requested information is somewhere on the Web at all). Needless to say that the search engine should be a lot faster, but speed alone is not an end by itself. For advanced queries of the kind mentioned above humans would surely tolerate a response time of up to a day if the results are useful and the search engine saves precious intellectual time.

The above emphasizes the effectiveness of searching in terms of finding good results. Equally important aspects are the completeness of the search in that all possible information sources are exhausted, and the efficiency of the search process. The Web as it is covered by the union of search engines today contains about 1 Billion (i.e., 10^9) documents with a total size of 20 Terabytes. In addition, however, there is a huge amount of interesting and relevant data behind Intranet gateways and other portals, typically but not necessarily in more schematic databases. This includes information sources such as CNN and other news providers, amazon.com which has descriptions and reviews of books, the US patent database, the library of congress, various climate data centers with lots of satellite images, and so on. This so-called "Deep Web" [4] is estimated to have 500 Billion documents with a total size of 8 Petabytes, and it is the fastest growing part of the entire web. So the "surface" Web that is in reach of today's search engines is less than 1 percent of all information sources. For 2010 I expect that a large fraction of the Deep Web will indeed be searchable in a unified manner, and that we will have found reasonable means to cope with the tremendous scalability problem that we are facing.

4.3 Research Avenues

4.3.1 The Great Synergy. Progress in information retrieval has been incremental, and we cannot expect a breakthrough in the next decade given the extreme difficulty of the intelligent information search problem. However, there are a number of trends each of which will gradually improve the state of the art, and all of them together will hopefully lead to great synergy and major advances in the quality of search results:

 The XML standard has the potential to foster semantically richer annotations of text documents, and query languages for XML provide powerful pattern matching capabilities on these annotations.

- Hierarchical ontologies organize documents in topics and can contribute to much more precise query formulation as well as query refinement in a way that is largely transparent to the user.
- To automate the building and maintenance of large-scale ontologies, *auto-matic classification* algorithms, based on statistical learning, can be lever-aged.

Note that these trends aim to automate the manual pre- and postprocessing that we have seen to be useful for better search results in the "Chernoff theorem" example of Subsection 4.1: XML pattern matching can replace the manual navigation in the environment of some promising URL, and ontologies with automatic classification should replace manually maintained, Yahoo-style, directories.

In the following subsections I will discuss the above research directions in more detail. I will also discuss implications on the architecture of a search engine under the perspectives of scalability and coverage of the "Deep Web".

4.3.2 XML. XML is a W3C standard and widely considered as the main driving force in the ongoing endeavor for uniform data exchange and integration of heterogeneous data across the entire spectrum from largely unstructured to highly schematic data. In an abstract sense, all data is uniformly captured by a graph with nodes representing XML elements along with their attributes and with hyperlinks being elements of a special type [1]. A variety of query languages have been proposed for searching XML data (see, e.g., [8,12,29]). These languages combine SQL-style logical conditions over element names, contents, and attributes with regular-expression pattern matching along entire paths of elements. The result of a query is a set of paths or subgraphs from a given data graph that represents an XML document or document collection.

As an example consider the following three XML documents about vacation destinations and traveling. Note that the element names and the structure of the XML documents vary slightly; so there is no universal schema. We may think of the three documents as coming from two different information sources, one for the left-hand column and another one for the right-hand column.

```
<region> Overseas
<region> Europe
<place> Sylt
                                            <sight> Townsville
  <location> Germany </location>
                                              <country> Australia </country>
  <br/>deach> sandy beach on the shore of
                                              <attractions>
          the North Sea </beach>
                                                <attraction> beach </attraction>
 <activities> dune hiking,
                                                <attraction> coral reef
              surfing </activities>
                                                </attraction>
 <season> summer </season>
                                                <what-to-do>
</place> </region>
                                                  <diving> scuba diving outside
                                                           the reef ? </diving>
                                                  <snorkeling> snorkeling in the
<region> Europe
                                                               coral reef ?
<place> Bernese Oberland
                                                  </snorkeling>
  <location> Switzerland </location>
                                                </what-to-do>
  <activities> skiing, hiking,
                                              </attractions>
                                              <time> all seasons </time>
              climbing </activities>
 <season> winter, summer </season>
                                            </sight> </region>
</place> </region>
```

It is important to note that the element names in the above example data are not chosen arbitrarily but rather comply, to a large extent, with certain standard terminology – or a domain-specific ontology as we may say. This is a key point of XML: although it merely provides a syntax, it is creating a big momentum in the industry and also the scientific community to organize data in a better way, for example, by using ontological frameworks. There are ongoing efforts to provide XML element names spaces and DTDs or even schemas for business-to-business e-commerce data (specifying the structure of purchase orders, invoices, etc.), publisher data (specifying book layouts), genome research and bioinformatics, mathematics (with element names such as <theorem> or <Abelian group>), and so on [17,38,41,52]

Element names that follow, to a large extent, certain terminological conventions, are semantically rich annotations that potentially capture the topic and content of a document in a clearer way than the document text alone and thus facilitate more precise queries. This is the opportunity for better search result quality that XML query languages aim to exploit. For example, a search for vacation destinations where you can swim in the summer can be expressed as follows (in the specific syntax of the language XXL [47], standing for "FleXible XML Search Language", but other XML query languages have very similar abstract syntax):

Select P From http://my.holiday.edu/allsights.xml Where region.(place | sight) As P And P.# LIKE "%swimming%" And P.#.season LIKE "%summer%"

Words put in boldface are keywords (e.g., Select), uppercase single letters are element variables (e.g., P), # is a wildcard for element paths of arbitrary length ≥ 0 , and dots denote path concatenation. The Where clause of the query is a logical conjunction of *path expressions* which specify regular expressions (using constructors *, +, ? for $\geq 0, \geq 1, \leq 1$ iterations, resp.) over elementary conditions on name or content of elements or attributes. Such elementary conditions include substring matching using the SQL-style LIKE operator and the wildcard symbol % for arbitrary strings within a single element's content. Element variables (e.g., P) are bound by the As clause to the end node of a path in the data graph that matches the corresponding path expression, and denote this node when used in other expressions.

Although the above query is a perfectly nice example that demonstrates the expressive power of XML query languages, it also shows the limitations of queries with a Boolean retrieval semantics, returning a set of qualifying results as it is usual in database query languages: the query result is the empty set, for none of the three given documents explicitly contains the word "swimming". This observation is not atypical for settings where the underlying data is partly or even largely unstructured such as the Web or comes from different, heterogeneous, information sources such as large Intranets. To make XML querying truly Web-enabled, ranked retrieval must be supported where the result of a query is a list of element paths (or subgraphs in general) in descending order of estimated relevance to the query. This kind of similarity search, in the spirit of information retrieval technology, is being pursued in a few, very recent language proposals [10,16,47]. In XXL this capability is provided by an additional elementary operator \sim that tests similarity of name or content of elements or attributes, using an underlying ontology (i.e., thesaurus in more mundane terms) that captures related terms and standard similarity metrics like the tf*idf formula. The above information demand can be better expressed by the following XXL query:

```
Select P
```

```
From http://my.holiday.edu/allsights.xml
Where region.~place As P
And P.#.(~swimming)? ~ "swimming"
And P.#.~season ~ "summer"
```

In this query the similarity operator \sim is used as both a unary operator on element or attribute names and a binary operator on element or attribute contents. For all kinds of elementary similarity conditions a similarity score is computed, and the scores from different conditions are combined using simple probabilistic arguments. The latter is based on postulating probabilistic independence between different conditions; better approaches that would capture correlations among element names and terms in element contents are subject of future research. The above query would return a ranked list with Townsville as the best match (i.e., the XML element bound to variable P that has the highest overall similarity score with regard to the query conditions) and Sylt as the second best match. This order is based on ontological similarity of element names which would rate "swimming" as more closely related to "snorkeling" than to "surfing".

More details on XXL can be found in [47], but note that this approach is still in a fairly early state, especially with regard to efficient implementation techniques. The key point that I wish to emphasize is that XXL and the few related projects reconcile two different search paradigms: the logical condition evaluation and pattern matching capabilities from database and XML querying, on one hand, and the similarity search with ranked results from information retrieval, on the other hand. It is this combination that bears a great potential for better search result quality. In contrast, traditional information retrieval and all current Web search engines completely miss the opportunity that arises from the more explicit structure and semantically richer annotations (i.e., element names) in XML documents.

4.3.3 Ontologies. Ontologies have been around in the AI community for more than a decade (see, e.g., [31,34,42]). The main goal has been to construct a universal knowledge base as a backbone for all kinds of automated reasoning. Consequently, the representation language was typically very rich, for example,

some form of description logic or higher-order logical assertions, but the scope and goal for *using* the ontology were not clearly defined. Today, we have a much clearer picture of what we would like to do with ontologies in the context of Web information retrieval. Furthermore, we should once again apply the engineering rule of "picking the low hanging fruit": relatively simple representations could already achieve most of the leverage within a reasonable timeframe, whereas perfect solutions often are elusive.

For example, the simplest kind of ontology would be a tree of topics (also known as categories or "concepts"): each topic could be characterized by a set of names (i.e., synonyms) for the topic, where the same name can appear under different topics (i.e., if it is a polysem) [15]. The edges of the tree would represent topic-subtopic (i.e., specialization) relationships. For example, a topic "science" would have subtopics "mathematics", "philosophy", "physics", etc.; "mathematics" would have subtopics "algebra", "stochastics", etc., and "stochastics" would in turn have finer-grained subtopics such as "statistical hypothesis testing" or "large deviation theory". Richer structures for ontologies are conceivable and not unlikely to be needed, for example, lattices of topics, but it is debatable whether a full-fledged logic-based knowledge representation language is worthwhile. Each topic in the ontology has its specific terminology, which could be captured in the form of "archetypical" sample documents or, more explicitly, as a set of specific terms with weights like tf and idf values or even correlation measures between terms; the latter can be computed from the sample documents. For example, we would expect terms such as "variability", "tail", or "bound" to have high weights within the subtopic "large deviation theory", and even "Chernoff bound" might be a term that is explicitly known in the ontology. In addition, with XML the description of a subtopic should also include typical element names or element paths along with statistical information about them.

Ontologies can, and often should, be domain-specific or even personalized in that they reflect the specific interests of an individual or community (see, e.g., [26,27,45]). Queries can be directed to a domain-specific ontology (just like we make use of domain-specific search engines already today) and would be enhanced by transforming them into a more precise form with richer annotations. For example, the search request for "Chernoff theorem" when issued to a mathematical ontology could be rewritten into the following query, specified in XXL because I ultimately expect all queries to be on XML data:

Select P

```
From http://my.math.ontology.edu/math.xml
Where #.math?.#.(~large deviation)?.#.theorem.~Chernoff As T
```

This obviously requires mapping the original query to the best fitting subtopic in the ontology and deriving the enhanced query from this subtopic's description. Likewise, queries that give some reasonable but not yet fully satisfactory results can be refined by having the user provide feedback and mapping the good results into the ontology to derive a refined query. All this can be done in a mostly implicit manner, with relatively little effort for the human user, through comfortable user interfaces. In fact, such relevance feedback is an old hat in information retrieval research, but has had amazingly little impact on Web searching so far. On the other hand, the combination with hierarchical ontologies and richer annotations in the form of XML elements does not seem to have been explored so far.

4.3.4 Automatic Classification. Automating the building and maintenance of a hierarchical ontology requires automatic classification, typically using some form of supervised, statistics-based, learning. So we start out with a seed of training documents that have to be classified intellectually. Good training sets could be derived from users' bookmarks or other carefully chosen, archetypical documents (e.g., by extracting positive results from users' "surf trails" in a semi-automatic manner). This way each topic can now be characterized by its specific distribution of terms, element names, paths, etc., or *features* in general.

The key building block then is a statistical classifier which needs to estimate the conditional probability that a new, previously unseen, document belongs to some topic given its feature vector. Here we can exploit Bayes theorem or more powerful mathematical machinery to derive these probabilities from statistical parameter estimates that we obtain from the feature distributions of the training sample. Documents are assigned to the topic to which they belong with highest likelihood. The simplest and most popular classifier of this kind is known as the Naive Bayes method, which makes a number of strong, simplifying assumptions such as postulating independence of term frequencies. Naive Bayes often performs not much worse than many of the more sophisticated methods; after all, we only have to estimate the odds that a document belongs to a topic versus not belonging there [32]. Nevertheless, better classifiers, most notably, methods based on support vector machines, are absolutely worthwhile to explore for (XML or text) document classification (see, e.g., [13,25] and the references there).

The quality of automatic classification is measured empirically by intellectually inspecting the fraction of correctly classified documents. Unfortunately, exeriments in this field are fairly limited and would have to be viewed as toy experiments relative to the sheer size and diversity of the Web. For example, a typical benchmark looks at 10,000 short newswire articles from Reuters, which belong to about 100 different categories, takes 75 percent of these articles as training data, and then automatically classifies the remaining 25 percent. Note especially that all documents really come from one of the 100 categories, which would not be the case (or would not be known) for highly diverse Web data. And even in such a relatively simple setting, the best classifiers are typically correct only for about 80 percent of the documents.

So there is work to do along these lines, and we need to explore better statistical learning techniques and intensify our efforts on large-scale experimentation. A critical aspect is also the selection of which features of a document should be used for the classifier. The simplest approach would just look at tf and idf values of individual terms, but term pairs or triplets could be used as well and often it is better to use only a small subset of "meaningful" terms as features to eliminate "noise" and to achieve scalability. The latter approach incurs subtle but critical difficulties with hierarchical classification: a term that serves well as a discriminator at one level of the ontology may be an inflationary and thus meaningless term at the next lower level. For example, the (frequent occurrence of the) term "theorem" intuitively is a good indicator for a document to belong to the topic "mathematics" (say rather than "humanities"), but it does not provide any glues for identifying the proper subtopic within mathematics. There is some preliminary work on how to determine whether a feature is discriminative or not, based on information-theoretic considerations [6]. However, this is limited to simple terms (i.e., normalized word occurrences) in plain text documents; the same question is widely open for XML elements or element paths.

In general, our understanding of these issues still seems to be in its infancy, and I would very much wish to see a new "master theory" of information content and relevance. Obviously, such a theory should leverage information-theoretic principles, but needs to eliminate the fairly strong, and often unrealistic, assumptions (especially with regard to feature independence) that have typically been made in the prior literature on information retrieval (including the perhaps most ambitious work [40]). Such a theory would be useful for reasoning about the "optimality" of a search result, even if it cannot eliminate the inherent uncertainty in vague queries and even if it is not practical (e.g., for complexity reasons) but could still serve as a yardstick against which real search engines could be systematically compared.

4.3.5 Deep Search. Current search engines do not reach the Deep Web because they rely almost exclusively on precomputed results: crawling the web and building up index structures, with query processing mostly boiling down to a few fast index lookups. For reaching data behind Intranet portals and for fully exploiting the capabilities of next-generation query languages such as XXL, more powerful "deep search" strategies are needed.

Index structures are still key for scalable query processing, but index lookups should only provide seeds for further automatic exploration in the neighborhood of promising URLs. Topic-specific crawlers [7] should be spawned from such URLs at the run-time of an individual query for "semantic" pattern matching according to the given XXL-style query. When coming across a portal behind which no direct crawling is allowed but which supports a search interface (e.g., sites such as CNN or amazon.com), a subquery should be dynamically generated and issued to the portal's local search engine. With XML and XML querying becoming ubiquitous standards, the most promising interface between the global search process and the local search engines would be in the form of a prevalent XML query language. To support the global search in generating the best possible subqueries, portals should describe their contents in terms of an XML schema or DTD and export also a local ontology along with statistical representations of the various subtopics.

The envisioned search method combines paradigms from today's (indexcentric) Web search engines and prototypical approaches for query processing in database federations. Note that similar multi-tier search algorithms are already in use by some metasearch engines (or search brokers) that generate specific subqueries for different underlying search engines (possibly including topic-specific engines) and (claim or aim to) take into account the thematic coverage and profile of the underlying information sources (see, e.g., [46]).

The biggest challenge in making this extremely flexible search process practically viable lies in the scalability aspect: coping with thousands of information sources and, ultimately, the entire "Deep Web". In this respect caching and prefetching are key technologies to achieve good performance on the Web. More specifically, speculative prefetching of data and asynchronous, speculative pre-execution of subqueries as well as query result caching would be intriguing approaches to hide latency and provide acceptable response times, but it is still a major step from existing techniques for data caching and prefetching (e.g, [30]) to the effective use of statistics-based speculation at all levels of a "deep search" engine.

5 Concluding Remarks

In this paper I have outlined three grand challenges for Web-based information systems: guaranteed performance, continuous availability, and intelligent search for information. None of these is truly new. The critical importance of predictable performance has been stressed also in the strategic report of the US President's Information Technology Advisory Board [48], zero-downtime Web services have been a high-priority goal for the last few years already, and the urgent need for self-tuning, "trouble-free" information servers has been recognized, for example, also in [2]. Finally the need for better search engines is something that everybody realizes almost every day. All three of my grand challenges are included, in some form or another, also in Jim Gray's list of important research goals presented in his Turing Award lecture [21].

The most elusive goal among the outlined challenges clearly is the intelligent search with guaranteed search result quality. This challenge has been around since Vannevar Bush's vision of a world knowledge device coined Memex (see, e.g., [21]) or may even be traced back to philosophers like Gottfried Leibniz. It may well take much longer than the next decade to build Web search engines that can truly find needles in haystacks with result quality as good as the best human experts could provide with infinite time resources. But ultimately it is the paramount importance of this problem that dictates tackling it as an absolutely top-priority grand challenge in computer science.

References

- 1. S. Abiteboul, P. Buneman, D. Suciu: Data on the Web From Relations to Semistructured Data and XML, Morgan Kaufmann, 2000.
- 2. The Asilomar Report on Database Research, ACM SIGMOD Record Vol.27 No.4, 1998.

- R. Baeza-Yates, B. Ribeiro-Neto: Modern Information Retrieval, Addison-Wesley, 1999.
- 4. BrightPlanet.com: The Deep Web: Surfacing Hidden Value, White Paper, http://www.completeplanet.com/Tutorials/DeepWeb/index.asp.
- 5. S. Brin, L. Page: The Anatomy of a Large Scale Hypertextual Web Search Engine, 7th WWW Conference, 1998.
- S. Chakrabarti, B. Dom, R. Agrawal, P. Raghavan: Scalable Feature Selection, Classification and Signature Generation for Organizing Large Text Databases into Hierarchical Topic Taxonomies, The VLDB Journal Vol.7 No.3, 1998.
- S. Chakrabarti, M. van den Berg, B. Dom: Focused Crawling: A New Approach to Topic-specific Web Resource Discovery, 8th WWW Conference, 1999.
- D.D. Chamberlin, J. Robie, D. Florescu: Quilt: An XML Query Language for Heterogeneous Data Sources, 3rd Int. Workshop on the Web and Databases, 2000.
- S. Chaudhuri (Editor): Special Issue on Self-Tuning Databases and Application Tuning, IEEE Data Engineering Bulletin Vol.22 No.2, 1999.
- W.W. Cohen: Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity, ACM SIGMOD Conference, 1998.
- F. Cristian: Understanding Fault-Tolerant Distributed Systems, Communications of the ACM Vol.34 No.2, 1991.
- A. Deutsch, M.F. Fernandez, D. Florescu, A.Y. Levy, D. Suciu: A Query Language for XML, 8th WWW Conference, 1999.
- S. Dumais, H. Chen: Hierarchical Classification of Web Content, ACM SIGIR Conference, 2000.
- C. Faloutsos: Searching Multimedia Databases By Content, Kluwer Academic Publishers, 1996.
- 15. C. Fellbaum (Editor): WordNet: An Electronic Lexical Database, MIT Press, 1998.
- N. Fuhr, K. Großjohann: XIRQL: An Extension of XQL for Information Retrieval, ACM SIGIR Workshop on XML and Information Retrieval, 2000.
- 17. Gene Ontology Consortium, http://www.geneontology.org.
- M. Gillmann, J. Weissenfels, G. Weikum, A. Kraiss: Performance and Availability Assessment for the Configuration of Distributed Workflow Management Systems, 7th Int. Conference on Extending Database Technology, 2000.
- Graduate Studies Program on "Quality Guarantees for Computer Systems", Funded by the German Science Foundation (Deutsche Forschungsgemeinschaft), Department of Computer Science, University of the Saarland, Saarbruecken, Germany, http://www-dbs.cs.uni-sb.de/~weikum/gk.
- J. Gray, A. Reuter: Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1993.
- J. Gray: What Next? A Dozen Information-Technology Research Goals, Technical Report MS-TR-99-50, Microsoft Research, Redmond, 1999.
- W.I. Grosky, R. Jain, R. Mehrotra: The Handbook of Multimedia Information Management, Prentice Hall, 1997.
- V.N. Gudivada, V.V. Raghavan, W.I. Grosky, R. Kasanagottu: Information Retrieval on the World Wide Web, IEEE Internet Computing Vol.1 No.5, 1997.
- G. Haring, C. Lindemann, M. Reiser (Eds.): Performance Evaluation: Origins and Directions, Springer, 2000.
- M.A. Hearst (Ed.): Trends and Controversies: Support Vector Machines, IEEE Intelligent Systems, Vol.13 No.4, 1998.
- J. Heflin, J. Hendler: Dynamic Ontologies on the Web, Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), 2000.

- M.N. Huhns, L.N. Stephens: Personal Ontologies, IEEE Internet Computing, Vol.3 No.5, 1999.
- J.M. Kleinberg: Authoritative Sources in a Hyperlinked Environment, Journal of the ACM Vol.46 No.5, 1999.
- D. Kossmann (Editor), Special Issue on XML, IEEE Data Engineering Bulletin Vol.22 No.3, 1999.
- A. Kraiss and G. Weikum: Integrated Document Caching and Prefetching in Storage Hierarchies Based On Markov-Chain Predictions, The VLDB Journal Vol.7 No.3, 1998.
- D. Lenat, R.V. Guha: Building Large Knowledge Based Systems, Addison-Wesley, 1990.
- D.D. Lewis: Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval, European Conference on Machine Learning, 1998.
- D. Lomet, G. Weikum: Efficient and Transparent Application Recovery in Client-Server Information Systems, ACM SIGMOD Conference, 1998.
- P. Mitra, G. Wiederhold, M.L. Kersten: Articulation of Ontology Interdependencies Using a Graph-Oriented Approach, 7th Int. Conference on Extending Database Technology, 2000.
- R. Nelson: Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modeling, Springer, 1995.
- G. Nerjes, P. Muth, and G. Weikum: Stochastic Service Guarantees for Continuous Data on Multi-Zone Disks, ACM Int. Symposium on Principles of Database Systems, 1997.
- G. Nerjes, P. Muth, G. Weikum: A Performance Model of Mixed-Workload Multimedia Information Servers, 10th GI/NTG Conference on Performance Evaluation of Computer and Communication Systems, 1999.
- 38. OpenMath Content Dictionaries, http://www.openmath.org/cdfiles/html/extra.
- 39. Oracle8i with Oracle Fail Safe 3.0, White Paper, Oracle Corporation, 2000, http://www.oracle.com/tech/nt/failsafe/pdf/ofs30db.pdf.
- C.H. Papadimitriou, P. Raghavan, H. Tamaki, S. Vempala: Latent Semantic Indexing: A Probabilistic Analysis, ACM Int. Symposium on Principles of Database Systems, 1998.
- 41. RosettaNet Partner Interface Processes, http://www.rosettanet.org.
- S. Russell, P. Norvig: Artificial Intelligence A Modern Approach, Prentice Hall, 1995.
- A. Silberschatz, M. Stonebraker, J. Ullman (Editors): Database Research: Achievements and Opportunities Into the 21st Century, ACM SIGMOD Record Vol.25 No.1, 1996.
- 44. A. Silberschatz, S. Zdonik, et al.: Strategic Directions in Database Systems Breaking Out of the Box, ACM Computing Surveys Vol.28 No.4, 1996.
- S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Mädche, H.-P. Schnurr, R. Studer: Semantic Community Web Portals, 9th WWW Conference, 2000.
- 46. A. Sugiura, O. Etzioni: Query Routing for Web Search Engines: Architecture and Experiments, 9th WWW Conference, 2000.
- 47. A. Theobald, G. Weikum: Adding Relevance to XML, 3rd Int. Workshop on the Web and Databases, 2000.
- 48. US President's Information Technology Advisory Committee Interim Report to the President, August 1998, http://www.ccic.gov/ac/interim/.
- G. Weikum, C. Hasse, A. Moenkeberg, P. Zabback.: The COMFORT Automatic Tuning Project, Information Systems Vol.19 No.5, 1994.

- G. Weikum: Towards Guaranteed Quality and Dependability of Information Services) (Invited Keynote), 8th German Conference on Databases in Office, Engineering, and Scientific Applications, 1999.
- G. Weikum, G. Vossen: Fundamentals of Transactional Information Systems: Theory, Algorithms, and Practice of Concurrency Control and Recovery, Morgan Kaufmann, 2001.
- 52. The XML Cover Pages, http://www.oasis-open.org/cover/xml.html.