The Computational Database for Real World Awareness

CompDB

Thomas Neumann
Technische Universität München

Proposal duration: 60 months

Summary:
Two major hardware trends have a significant impact on the architecture of database management systems (DBMSs): First, main memory sizes continue to grow significantly. Machines with 1 TB of main memory and more are readily available at a relatively low price. Second, the number of cores in a system continues to grow, from currently 60 and more to hundreds in the near future. This trend offers radically new opportunities for both business and science. It promises to allow for information-at-your-fingertips, i.e., large volumes of data can be analyzed and deeply explored online, in parallel to regular transaction processing. Currently, deep data exploration is performed outside of the database system which necessitates huge data transfers. This impedes the processing such that real-time interactive exploration is impossible. These new hardware capabilities now allow to build a true computational database system that integrates deep exploration functionality at the source of the data. This will lead to a drastic shift in how users interact with data, as for the first time interactive data exploration becomes possible at a massive scale.

Unfortunately, traditional DBMSs are simply not capable to take full advantage of the improved hardware. Traditional techniques like interpreted code execution for query processing become a severe bottleneck in the presence of such massive parallelism, causing poor utilization of the hardware. I pursue a radically different approach: Instead of adapting the traditional, disk-based approaches, I am integrating a new just-in-time compilation framework into an in-memory database that directly exploits the abundant, parallel hardware for large-scale data processing and exploration. By explicitly utilizing cores, I will be able to build a powerful computational database engine that scales the entire spectrum of data processing – from transactional to analytical to exploration workflows – far beyond traditional architectures.
Section A. The Research Proposal: Extended Synopsis

Introduction

The goal of this project is to enable users to perform near real-time analysis and exploration of complex and large databases by exploiting modern hardware. For this purpose I propose to develop a computational database system that integrates all data processing tasks from transactional scripts to analytical SQL queries to exploratory workflows – in one system on the same most current (i.e., transaction-consistent) database state. In this sense I want to turn the database system into a comprehensive data science platform.

Engineering such a visionary system in a five-year time frame is only feasible by building upon my existing HyPer code base. At its current stage, HyPer is widely viewed as the most efficient in-memory database system for transactional OLTP as well as for analytical OLAP processing. I achieved this by developing many novel disruptive architectural changes, most notably the separation of transactional from query processing via hardware supported virtual memory snapshots [11] and the pioneering just-in-time JIT compilation framework [12]. Unlike traditional databases that rely on interpreted code execution, I developed a compiler for SQL and SQL scripts that generates machine-independent LLVM assembly code from highly optimized query plans [10]. HyPer is the first database system that exhibits such a holistic compilation approach.

In the proposed CompDB project I intend to convert HyPer into a full-fledged computational database that supports the entire data processing functionality (transaction, analytics, exploration) in one system on the same database state. This vision is visualized in Figure 1. The checkmarked functionality (transactions √ and analytics

![Figure 1: Architectural Sketch of the Computational Database](image-url)
is already in a mature state and can be built upon. In particular, I intend to re-engineer and vastly expand the just-in-time compilation framework in this research project. It will become the basis for providing multi-lingual support for expressive exploratory languages, as exemplified on the left-hand side of the figure. The compilation paradigm is the perfect basis for the multi-lingual and multi-modal (meaning data is viewed and processed in multiple representations) exploratory processing because it can provide data access in multiple views without actually materializing the views.

Deep data exploration typically involves multi-step workflows that successively process data to obtain insights [1]. In the proposed project, these processing pipelines are first optimized at the logical level and then compiled into executable code. The compilation framework of HyPer is visualized by the cog wheels that represent pre-defined operators that can be implemented in a high-level systems programming language like C++. The gear train that represents the LLVM code is generated for the application specific exploration scripts. This code instantiates and adapts the pre-defined operators and it realizes the data flow between successive operators (metaphorically, this code drives the sequence of cog wheels). At the beginning of such a processing pipeline the database is typically accessed by highly optimized scan-operators that utilize the processor support for SIMD vector processing. The SIMD vectorization optimizes the pre-selection of data items by pushing selection predicates all the way down into the scan code. This data retrieval is usually followed by relational algebra operators, like the exemplified join-operator \( \sqcap \), to combine, group and aggregate, or union the data of interest. Subsequent processing for data exploration typically involves statistical and relational algebra operations. My current implementation of the HyPer system is already fully operational for transactional and query processing. In this context I have demonstrated that my compilation framework achieves unprecedented performance, denoted “clock-speed processing speed” in the architectural sketch. The goal of the current project is to expand the functionality of the compilation framework to the new dimension deep data exploration while retaining the high performance. This will lead to a system supporting not only “information at your finger tips” (as propagated by SAP founder Hasso Plattner [13]) but to “insights at your finger tips”.

**State of the Art**

Currently, data processing is carried out in multiple dedicated systems: one for OLTP processing from which the data is extracted and loaded into a data warehouse for analytical processing and yet another system, such as map-reduce or R, for data exploration. The high volume transactions with short duration are executed on the primary database system. This is called online transaction processing (OLTP). At regular intervals (e.g., every night) the recent changes are extracted from the primary database, transformed if needed, and then loaded into a secondary database called the data warehouse. Now the long running analytical transactions (OLAP) are executed inside the data warehouse and not inside the OLTP system. As the two databases are decoupled except for this regular extract-transform-load step (ETL), the OLAP transactions no longer block the OLTP transactions. For data exploration, the data is usually transmitted to yet another compute infrastructure, like a distributed map-reduce cluster, or the R statistical programming system. The data redistribution delay alone impedes the performance of such a data staging architecture and renders near-real time exploration impossible. In addition, the currently used systems either do not scale to very large data volumes (e.g., R) or they require multiple redistributions while processing a complex exploration pipeline (e.g., multiple cleansing steps for noise eliminations [8]).

Besides the performance penalty, this task separation and its inherent data staging induces a severe consistency problem. Originally, all transactions were executed inside the same database system. This caused performance problems, but had a well defined semantics. All transactions saw the data as it is. With data warehousing, the performance problem goes away, but now the OLAP transactions see old data, namely the database instance as it was at the timepoint of the last ETL step. This is very unfortunate, as now the data that is available for analysis and exploration lags behind the real world. For example, in traditional business data warehousing it is usually not possible to analyze the sales/production/stock as of today, but only as of yesterday. In large companies with complex data warehouse installations the delay can be even larger as, sometimes, it is only possible to analyze the state of the company as it was last week. This severely violates real world awareness, as it limits the ability to react to recent changes. The implications of this can be seen even more severely in areas like e-science [6], where new information has to be ingested by a DBMS at a very high rate, but where this data can only be
analyzed and explored with a significant delay of minutes or even hours.

For many applications this high delay is very undesirable. Hasso Plattner of SAP demands to guarantee “information at your fingertips” [13], i.e., users should be able to analyze in near real-time the current data as it is, without any delay (e.g., for recommendations [9]). Similar for other commercial database vendors, John Metzger, CTO of IBM Netezza, stated that users try to push the ETL delay down towards real-time business intelligence. And this is very understandable, as it is a huge difference if one can evaluate complex analytical queries and exploratory workflows about the world as it is now, as opposed to if one can only gain insights about the world as it was quite a while ago. Being able to analyze the “real world” in the presence of high update rates would be a huge step forward, and would change the way users would interact with their transactional data. Business software experts claim that such a functionality would cause a similar revolutionary change in business processes as the initial digital data processing created half a century ago. Likewise, many science disciplines resort to data science which benefits tremendously from interactive data exploration capabilities.

Historically, and caused by the hardware and software architecture, data staging and warehousing was the only viable solution when combining high update rates with complex analytics. But hardware is changing: Today a machine with 1TB of main memory costs about 30,000€. And projects like RAMCloud at Stanford [14] promise to deliver up to 1PB of main memory in the near future. This makes it possible to keep most transactional data in main memory, which severely impacts the architecture of a DBMS, as pioneered by MonetDB [4]. But now the software architecture has to change as well in order to exploit this massive change in hardware. Existing DBMSs cannot be adjusted to do so – as also observed by the NoDB developers who propose to circumvent the database altogether [2]. We need a disruptive change in DBMS architecture. Extensive use of main memory allows for techniques that were not feasible in disk-based systems. Another important trend is the continuing increase in the number of available cores. Today servers have 60 cores and more, and one can expect to have hundreds or even thousands of cores in the very near future. This brings massive parallel processing power, which again allows for techniques that were infeasible in previous times. On the other hand keeping such a large number of cores occupied is a non-trivial issue; concurrency control is a major challenge here.

I propose to develop and implement a new computational DBMS architecture that embraces these hardware trends, and uses them for scalable processing of the entire spectrum, from transactional to analytical processing to deep data exploration pipelines on the same data – not on “stale” copies of the data. By exploiting the hardware characteristics I will be able to achieve significantly better performance than any known software solution. The unified computational database and data science system CompDB will perform a “triathlon” of OLTP and OLAP and Exploration and achieve world record performance in all three disciplines not in sequence but in parallel. Benchmark results of the current state of my HyPer-system for OLTP and OLAP workloads have shown that this is not wishful thinking but indeed feasible. This capability of performing complex analytics and deep exploration on up-to-date data under high update rates is a dramatic improvement in the capabilities of a DBMS, and will change the way users interact with their data. Note that there is little chance of adapting a traditional DBMS towards this goal: The hardware trends have a very severe impact on how a DBMS should be constructed, and a more radical change is required to exploit the hardware to its full potential.

Impact of Hardware Trends

The most visible of the current hardware trends are the continuing increase in main memory and the increase in aggregated compute power due to multi-core parallelism. Traditionally, database performance has been largely determined by the available I/O performance, and DBMSs invested a lot of effort to minimize I/O. In particular random disk I/O is very expensive, as there the disk head has to be moved repeatedly across the platter. With increasing main memory sizes the amount of disk I/O is greatly reduced. A commodity server, e.g. from Dell, with 1TB of main memory currently costs about 30,000€; even a server with 6 TB is readily available. As a consequence, most of the actively accessed transactional data will reside in main memory. Archives with historical data might go beyond this size for now, although even this will probably change in the near future as memory sizes continue to grow. This can be seen by a simple back-of-the-envelope calculation: One of the largest commercial enterprises, Amazon, has a yearly revenue of about 100 billion Euros. Assuming that an individual order item costs about 15€, and assuming 54 bytes of storage per item – as specified in the TPC-C benchmark – the yearly data volume for orders is about 360GB. Now Amazon might store some additional data,
but it is very reasonable to assume that the data volume created by orders is well less than 750GB per year, which easily fits into a few TB of main memory. And most companies are much smaller than Amazon.

Assuming that most of the active data resides in main memory greatly simplifies the design of a DBMS. For example the system can afford to use pointers, which is naturally and very cheap in main memory, but which would cause very expensive random I/O in a disk-based system. Furthermore, main memory allows for using techniques that are not available on disks. One major property of main memory is the concept of virtual memory, which offers a hardware-supported linear address space to each individual process. As prior work, I developed the HyPer DBMS [9], which uses this mechanism to implement transaction isolation. The main idea in HyPer is that long-running transactions fork the database state, that is, they create a virtual copy of the database. This copy remains stable over the whole execution time of an analytics session, and effectively isolates the long-running queries from the very frequent short-running transactions without any software control mechanisms. As this copy is realized via virtual memory, it is very cheap. Initially, the copy is purely virtual, i.e., the DBMS creates a new process for the long running transactions, but does not copy any data. Now when a transaction in the main database updates a data item that is also visible to the long running transactions, the hardware MMU creates a trap (as a shared memory page is about to be modified), and the OS creates a copy of the updated pages. Thus, memory pages that are actively modified are copied on demand, de-coupling the long-running transaction from the other transactions. However pages that are not modified (which is usually the vast majority in a database) are shared, and cause not costs. As this whole checking and replication mechanism is supported by the hardware, is is extremely fast, much faster than a pure software solution. Using these techniques, HyPer has achieved world-record OLTP and OLAP performance concurrently [9], i.e., it matched the performance of specialized OLTP and OLAP systems within one combined workload. This is a dramatic improvement over the state of the art, and was possible because HyPer does not treat main memory like a fast disk, but really exploits the capabilities of main memory.

In this \textit{CompDB} project I will build upon the snapshotting mechanism to allow exploratory workflows on the most current state of even very large in-memory databases. However, it is foreseeable that the extensive usage of main memory will become more complex in the future. While main memory sizes continue to grow, the nature of main memory changes. Modern servers with very large amounts of main memory tend to be NUMA architectures. While it is still possible to access the whole memory from everywhere in these machines, some memory accesses will be cheaper and some will be more expensive. This requires considerable thoughts about the processing engine architecture because a server has become a network of many cores and NUMA-nodes in itself. The question is not only how to layout the data to maximize NUMA locality, but also how to arrange the exploratory workflow processing such that synchronization and transfer across NUMA nodes it minimized.

The second hardware trend that has a very severe impact on the computational DBMS architecture is the growing number of cores from currently 60 to hundreds. Keeping such a large number of cores busy is a challenging task. Even more so in a computational DBMS that has to deal with data skew such that some parallel tasks take much longer than others. The synchronization and load balancing costs in such a scenario can easily dominate the execution costs, and today’s DBMSs simply do not scale beyond a few dozen nodes/cores. This development requires a major rethinking in how to parallelize and synchronize a DBMS, as otherwise the performance of DBMSs will stagnate. The performance of an individual node is not improving significantly any longer, a DBMS must scale by massively parallelizing its tasks.

**Goals and Challenges of the Project**

The most prominent data exploration systems, often coined Big Data systems, are geared toward huge data sets that are found in multi-national companies like Google, Facebook, etc. In contrast to these projects, like Hadoop, Spark, Flink [3], etc., the goal of my project is to provide data exploration within the database that runs on a powerful server or a moderate cluster of powerful servers; not on hundreds or thousands of distributed low-end workstations in the cloud. Therefore, processing in \textit{CompDB} is CPU-bound rather than limited by the costly data transmissions. \textit{CompDB} is designed to serve the end-to-end data processing needs of typical database usage companies or research organizations. Other than the Big Data platforms my system is geared towards (1) near-real time performance (2) on the most current transaction-consistent database state for truly interactive data science. Some important architectural aspects of \textit{CompDB} are now briefly surveyed.
Multi-Lingual Workflows  As user-defined code fragments (so-called UDFs) prevail in data exploration workflows I want to allow them to be defined in a language of choice, e.g., in R, Julia, Python, etc. My compilation framework basically allows to incorporate any of the many languages that provides an LLVM compiler, even mixing code modules from multiple languages. Using LLVM as intermediate representation it will be possible to inspect the behavior of those UDFs for optimization purposes rather than treating them as black boxes.

Multi-Modal Database Views: Application-Specific Representation  The relational model has gained dominating acceptance because of its simple, generic data structuring. However, the pure and simple format is not necessarily the best representation for computationally complex applications that prefer graph, hierarchy or object representations. However, with the emerging main-memory database systems a coexistence of different models or views of the same data becomes possible. We call this the multi-modal view database. Here, data is physically represented in one (i.e., the relational) storage format. But applications can transparently operate on a different representation, i.e., on a view of the data. Thus, in the multi-modal view database the same data can be viewed in different logical models, for example in the standard relational row-model, or in a (business) object model representation, or in a graph model. It should be re-emphasized that the model constitutes only a virtual view of the data, not its physical representation. The interoperability is enabled by the just-in-time compilation approach whose basis I already developed in the HyPer-system. All operations, SQL or object operations, will be compiled into LLVM code that operates on the one physical representation of the data, thereby replacing the conceptual view [5].

Alleviating the Impedance Mismatch between SQL and Statistical Languages  Formulating complex explorations, such as machine learning tasks, in SQL is often quite cumbersome [16]. The user interface of CompDB will merge SQL with conventional (functional) languages in order to explicitly control the data flow within and between workflow UDF tasks. In particular, the interface will provide global state variables and iteration control as many algorithms in statistics and ML have iterative nature until globally observable convergence criteria are satisfied. By conceptually integrating these language paradigms, I can alleviate the often-cited impedance mismatch between set-oriented SQL and fine-grained record-at-a-time processing in imperative languages.

Holistic Optimization of Exploration Workflows  Having a common intermediate representation in the form of LLVM it becomes possible to analyze the behavior (selectivity, CPU usage, memory consumption). This knowledge allows a holistic optimization from fine-grained compile-level strategies (e.g., generating vectorized SIMD code) to classical access path selection (e.g., usage of an index) to task allocation to particular servers or co-processors (like GPUs) within the database cluster. A particularly effective compile-level optimization is the generation of pipelined code – possibly even across UDFs – that achieves maximal data locality in a NUMA or cluster environment. At the logical level my group has achieved impressive optimizations by converting record-at-a-time computations into set-based batch processing for graph exploration [15]. Also, identifying common sub-solutions and materializing their intermediate results in a workflow promises huge gains.

Parallelization and Distribution  We distinguish between parallelization in the small and distribution of the execution that constitutes parallelization in the large. For parallelization within a node we have compiler-controlled SIMD vectorization, NUMA-local thread allocation and workload management among parallel threads as effective means for speeding up compute-intensive tasks. Distributing the computation across nodes in a cluster has to detect collocated data patterns in order to reduce costly data transfer. For unavoidable data transmission the latest communication infrastructures, such as RDMA over InfiniBand, will be exploited.

Funding  To cope with this ambitious project, I request funding for four research assistants, who will concentrate on one of these issues each. They will be assisted by two additional research assistants funded by the University.
Additional References


Section B. The Principal Investigator: Curriculum Vitae

Personal Data

Name: Thomas Neumann  
Birth: 1977-02-22 in Köln, German nationality  
Address: 85748 Garching, Brauneckweg 26, Germany

Academic Positions

2010 -  Professor (W2), Technische Universität München, Database Systems  
2010  Professorship offer at FU Berlin (rejected)  
2009 and 2015  Visiting Researcher, Microsoft, Redmond  
2008  Temporary Professor (Lehrstuhlvertretung), Universität Heidelberg  
2005 - 2010  Senior Researcher, Max-Planck-Institut für Informatik, Database Systems Group  
2001 - 2005  Research Assistant, Universität Mannheim, Database Systems Group

Education

2010  Habilitation in Computer Science, Universität des Saarlandes  
2005  Doctoral degree (Dr. rer. nat.) in Computer Science, Universität Mannheim (grade summa cum laude)  
2001  Diploma in Computer Science (Dipl. Wirtsch.-Inf.), Universität Mannheim (grade 1,4)  
1997 - 2001  Studies in Computer Science, Universität Mannheim  
1996  Abitur, Städtisches Gymnasium, Wesseling (grade 1,3)

Funding

Basically all funding of more than 2 million Euros I raised at TUM is related to the HyPer project that started in 2010 and builds a main memory DBMS for mixed workloads OLTP and OLAP [8, 9, 11]. There is no functional overlap between the existing funding and the proposed CompDB projects; however, the proposed project will build upon the HyPer code base that was partially supported by the previous/current funding.

Teaching

I regularly teach the entire spectrum of lectures in the database area. At TUM I have repeatedly taught the two basic compulsory database lectures “GDB: Grundlagen Datenbanken” and “ERDB: Einsatz und Realisierung von Datenbanksystemen” for 850 and 250 students, respectively, as well as the advanced database courses on “Query Optimization,” “Implementing Databases on Modern CPU Architectures,” and “Transaction Management”. I am also active in the joint TUM, LMU, Uni Augsburg elite student program “Software Engineering.” I also initiated and coordinate the new TUM Master’s program “Data Engineering and Analytics”.

Database Community Services

Regular (basically every year) program committee member of the following conferences: VLDB, SIGMOD, ICDE, EDBT, Eurosys. Reviewer for all major DBMS journals.  
General Chair of the VLDB 2017 conference in Munich.  
Associate Editor of the VLDB Journal.
Scientific Leadership Potential

One of the most visible first results after graduation was my work on the RDF-3X systems, which is a database system for graph-structured RDF data. By using a combination of indexing techniques and my query optimization techniques it achieves excellent performance when querying and processing large-scale RDF data. It is still (almost a decade after engineering it) widely regarded as one of the fastest, if not the fastest, DBMS for RDF processing. This is also evidences by its adoption in Daniel Abadis Hadoop project [presented at VLDB2011], as Abadi had previously advocated his own RDF database system on top of a column store, for which he got the VLDB best paper award [2007] and the ACM SIGMOD dissertation award [2008]. Being open source, it has been used as basis for further research at several universities both within Europe and the US. My RDF-3X related publications have well over 1000 citations, and are still actively used and very visible within the community.

My second very successful area of research is that of query optimization. After my PhD I developed a new class of graph-based algorithms for the classical join ordering problem, and showed how they can be used to handle queries of arbitrary size and complexity. This greatly improved optimization time relative to the state of the art, both asymptotically and in absolute terms, and has great practical relevance. Formerly, due to the exponential complexity, query optimization was constrained to queries of up to a dozen or so joins. My new breakthrough moved this barrier to hundreds of joins by relying on a very different, more efficient enumeration technique. The commercial Sybase DBMS incorporates a variant of the algorithm, and Microsoft invited me to study its application within SQL Server. As I continue to collaborate with my former PhD advisor in this area up to this date, most of my publications in this area are missing in my list, as discouraged by the ERC Guidelines.

After moving to Munich in 2010 I started working on the HyPer DBMS which I view as my “Opus Magnum”. HyPer is a main-memory based relational DBMS for mixed OLTP and OLAP workloads. In Turing-award winner M. Stonebraker’s terminology it is a New-SQL database system that entirely deviates from classical disk-based DBMS architectures by introducing many innovative ideas like virtual memory based transactional snapshots and machine code generation for data-centric query processing, leading to exceptional performance. Based upon the two major foundational publications in ICDE2011 and VLDB2011 it has quickly gained significant international visibility, including an invited keynote at BIRTE2010, several talks at commercial database vendors (e.g., HP, Oracle, SAP, IBM, Microsoft, etc.), and a demonstration at VLDB2011. The work on HyPer is still continuing and each year thereafter, I engineered and published new concepts of the HyPer database system at all major and highly selective database conferences (SIGMOD, VLDB, and ICDE) – including a best paper award winning entry in ICDE 2013. Even for the 2016 editions of these conferences, four HyPer papers were accepted (SIGMOD, 2xVLDB, ICDE) and some more are in the pipeline. HyPer is viewed by many as the most innovative academic database system project. My leadership potential in the international database community is best reflected by my winning the VLDB Early Career Innovation Award 2014 which is granted to only one pioneering database researcher per year by the VLDB Endowment. I received the award for my comprehensive expertise in “Architecting High Performance Database Engines.” The other recent award winners include Prof. Daniel Abadi of Yale University (2013) and Prof. Chris Re (Stanford University) – so far I am the only European to win this award in the traditionally US-dominated field of databases.

Overall I have published over 80 peer-reviewed, international publications with almost 3000 citations, I was part in ca. 20 conference program committees, am Associate Editor of the VLDB journal, and reviewed a large number of journal articles. More on the practical side I have a lot of experience with designing and implementing database management systems, and I have demonstrated the impact of a superior architecture very successfully with both RDF-3X and HyPer. As such, I am very familiar with the challenges and tasks associated with this ambitious proposal that requires enormous system software engineering efforts. Given my track record in database systems engineering, I am very confident that I will be able to successfully architect and lead the research development project.

I am currently supervising five doctoral students working on RDF-3X and HyPer. I coached two teams assembled from my PhD students for the ACM SIGMOD programming contest: In 2013 it finished second and the 2014 team won this prestigious contest. I would use the grant to hire four additional research assistants in order to design and build a new computational database system architecture for deep data exploration on new hardware. The existing HyPer code base which achieves unprecedented performance for transaction and SQL query processing will serve as a basis for this new line of work.
**Funding Summary**

**On-going Grants**

<table>
<thead>
<tr>
<th>Project Title</th>
<th>Funding Source</th>
<th>Amount (Euros)</th>
<th>Period</th>
<th>Role of PI</th>
<th>Relation to ERC Proposal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid OLTP/OLAP main-memory DB</td>
<td>DFG NE 1677/1-1</td>
<td>610,120€</td>
<td>2010-2016</td>
<td>PI</td>
<td>code base for CompDB</td>
</tr>
<tr>
<td>Hierarchy support</td>
<td>SAP</td>
<td>543,500€</td>
<td>2014-2016</td>
<td>co-PI</td>
<td>no direct rel.</td>
</tr>
<tr>
<td>Graph Analytic and Distributed Query Eval.</td>
<td>Oracle Labs</td>
<td>ca. 315,000€</td>
<td>2014-2017</td>
<td>co-PI</td>
<td>code base</td>
</tr>
<tr>
<td>LDBC Linked DB Council</td>
<td>EU 317548</td>
<td>240,120€</td>
<td>2014-16</td>
<td>PI</td>
<td>no rel-ship</td>
</tr>
<tr>
<td>Software Campus 4 PhD projects</td>
<td>BMWi</td>
<td>410,903€</td>
<td>2013-17</td>
<td>supervisor of PhDs</td>
<td>contribution to code base of HyPer</td>
</tr>
</tbody>
</table>

**Overlap with the ERC Proposal**

Basically all funding of more than 2 million Euros I raised at TUM is related to the HyPer project that started in 2010 and builds a main memory DBMS for mixed workloads OLTP and OLAP [8, 9, 11]. There is no functional overlap between the existing funding and the proposed CompDB projects; however, the proposed project will build upon the HyPer code base that was partially supported by the previous/current funding.

**Grant Applications**

None
Section C. Early Achievement-Track-Record

The publication list below contains selected international peer-reviewed publications of the PI. As suggested by the ERC guidelines any collaborations with the PI’s former PhD advisor were omitted, even though several of these collaborations were many years after the PhD. This post-doctoral collaborations, which occurred after the PI moved to MPI and which were not derived from the dissertation, included significant breakthroughs in query optimization. In particular the new join ordering algorithm developed by the PI in collaboration with his former advisor significantly speeds up query optimization, both absolutely and asymptotically, and has made it into the Sybase commercial DBMS.

Citation counts below are derived from Google Scholar (scholar.google.de/citations?user=xSDfDpsAAAAJ when available. Overall, my publications have been cited 2850 times.

Journals and Most-Selective Conferences in the Field (10 selected Papers)


Workshops


Book/Book Chapters


Patent


Prizes and Awards

• VLDB Early Career Innovation Award for “Engineering High-Performance Database Engines, 2014 (awarded at the VLDB International Conference in Hongzhou, China with a plenary presentation of my work in front of ca. 800 delegates)

