DATAWARE VENTURES

WHITE PAPER:

Field Specialization

OVERVIEW

DBMS vendors want to improve the performance—specifically, execution speed—of their products, to meet the increasing needs of their customers and remain competitive. Existing approaches for performance improvement, such as domain-specific architectures, query compilation, and JIT compilation, are costly for vendors to implement in both time and money.

Field specialization is a patented new approach for DBMS performance improvement that addresses these problems. It enjoys the following advantages:

- ✔ Significant performance improvement
- ✔ Requires only small, localized changes to existing DBMS code
- ✔ Quick time-to-market
- ✔ Flexibility in selecting where to specialize code
- ✔ Granular control, enabling applications to be customized to individual customer needs
- ✔ Equally applicable to legacy DBMS code and to cutting-edge technology, such as cloud-based, main-memory, and MPP DBMSs
- ✔ Complementary to other performance improvement approaches
- ✔ Applications built on top of DBMS require no changes
- ✔ No action required for end users

While field specialization is applicable to many different software application domains, this document focuses on the application of field specialization to DBMSs.

As an ongoing proof of concept, field specialization was applied to an industry-leading DBMS, PostgreSQL (PG) 9.3, and performance was evaluated using the industry-standard TPC-H and TPC-DS benchmarks. The total runtimes of the top five queries were as follows.

Benchmark	PG 9.3	Field-Specialized PG 9.3	Field Specialization Advantage
TPC-H	105sec	51sec	2.04X speedup
TPC-DS	3,063sec	1,001sec	3.06X speedup

Table 1. *Speedup provided by field specialization*

Field specialization thus currently provides 2X–3X speedup in PostgreSQL 9.3 for these two oft-used benchmarks.

This white paper, intended for technical readers, describes:

- The concept of field specialization
- How Dataware Ventures' products can support the DBMS vendor in the field specialization process
- How other performance improvement approaches compare
- Performance improvements possible with field specialization

THE DBMS VENDOR'S CHALLENGE

DBMS vendors receive constant pressure from their customers, the DBMS end user, to improve the performance of their DBMS products. Interviews with CIOs from major companies revealed the following:

- "Performance is always an issue."
- "Reducing the mean time of queries is very important."
- "We invest hundreds of thousands of dollars each year to speed up queries."
- "Everyone is desirous of 2X performance improvement."
- "Queries have to run fast."
- "We need to run apps faster during the day."

And the list goes on. Clearly, DBMS vendors are under the gun to deliver faster products before their competitors do.

The main question DBMS vendors ask themselves, then, is: "How?" Several approaches exist to improve DBMS performance, from optimizing code piece-by-piece to rearchitecting the entire DBMS to incorporate a new technology. Each approach has its advantages and disadvantages, but all are costly in terms of time and money.

Field specialization is a new approach for improving DBMS performance that is complementary to these other approaches while being easy to deploy and not requiring rearchitecting the DBMS.

WHAT IS FIELD SPECIALIZATION?

A DBMS is designed to be generic: to handle any workload (i.e., data, schema, queries) that a user defines. This generality incurs a performance cost. For example, fetching a row requires frequently looking up schema information to get e.g. the number of columns, their type, and their size, even though the schema information is usually static. Any particular workload has information that is specific to it, such as this schema information. The key insight behind field specialization is that workload-specific information induces runtime invariants in the DBMS code that can be exploited to improve DBMS performance.

Field specialization is the process of inserting *spiffs* into DBMS code so that the DBMS can specialize itself at runtime to exploit runtime invariants.

A spiff, which stands for ***sp***ecializer ***i***n the ***f***ield, is code that dynamically creates specialized code at DBMS runtime. The specialized code is both smaller and faster than the original unspecialized code.

Field specialization gets its name from the fact that the code is specialized "in the field," i.e., after the DBMS has been deployed and is running at the end user's site. A spiff uses the actual value of a runtime invariant—which can only be known at runtime—to dynamically produce code that is specialized to that particular value of the runtime invariant.

How Spiffs Improve Performance

Current technology trends, including main memory databases and hybrid SSD secondary storage, shift system performance drivers away from disk performance and towards in-memory execution parameters. Performance improvements from the specialized code created by spiffs result primarily from a reduction in the number of memory references (including instruction and data references) and cache misses. As these shifts in technology continue, the performance impact of field specialization will be magnified beyond what we obtain today.

A Simple Example

- A database has a "sales" table with 42 columns. The number of columns doesn't change during the execution of a query. Thus, the number of columns is an example of a runtime invariant.
- The DBMS contains a small generic routine to extract individual column values from an input row. The routine performs a loop over the columns.
- In contrast, a spiff will create specialized code at DBMS runtime. This specialized code is a result of the spiff performing
  - constant folding,
  - loop unrolling 42 times, and
  - memory access coalescing to reduce cache pressure.

  Thus, this spiff replaces a generic piece of code with code that is highly specialized to values only known at table creation time.
- Each table in the database will have its own specialized code based on its number of columns. The specialized code for each table will be created by this same spiff, at DBMS runtime.

Opportunities Everywhere

Runtime invariants of many kinds can be identified during a spectrum of times, from database creation to query execution. Field specialization can take advantage of all of these kinds of invariants.

Individual spiffs each contribute a performance improvement. An individual spiff may contribute a mild improvement across the board, or a more extreme improvement in specific but commonly-encountered cases. In concert, these spiffs provide a significant performance improvement to the DBMS.

DATAWARE'S SOLUTION

Dataware offers an end-to-end solution to support the field specialization process. The solution consists of two main components:

- *The Spiff Runtime Environment* (SRE) provides an API to manage spiffs and the specialized code they create at DBMS runtime.
- *The Spiff Toolset* provides static and dynamic program analyses to identify runtime invariants in the DBMS code and supports the creation of spiffs for those invariants.

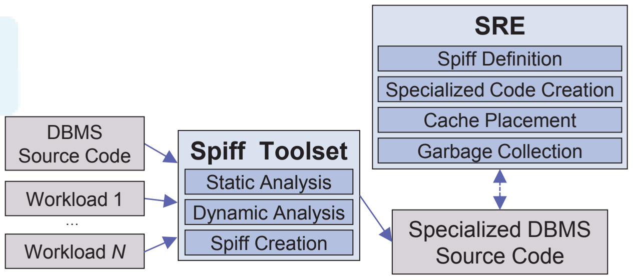

Figure 1. *Dataware's solution*

The Spiff Runtime Environment (SRE)

The SRE is a library that the DBMS vendor will use to manage spiffs and the specialized code they create.

- ✔ Supports the definition of spiffs at DBMS development time. The SRE allows spiffs to be defined as simple code annotations around existing DBMS code.
- ✔ Supports spiffs in the creation of specialized code at runtime
- ✔ Supports managing the specialized code at DBMS runtime, including optimized cache placement, dynamic linking, and garbage collection

The SRE makes it easy to define spiffs in the DBMS code and allows the DBMS to easily create code dynamically at DBMS runtime.

The Spiff Toolset

The Spiff Toolset supports the DBMS developer in identifying runtime invariants, creating spiffs that exploit those invariants, and inserting SRE API calls to support the spiffs and specialized code at runtime.

- ✔ Provides sophisticated static analyses of the DBMS source code to identify all possible runtime invariants
- ✔ Provides dynamic analyses of the DBMS on a given workload to rank the runtime invariants in terms of cost and benefit
- ✔ Supports the creation of spiffs for the most cost-effective runtime invariants

The closer an end user's workload is to the workload used in the dynamic analysis step, the better the performance improvement will be for that user. Many representative workloads can be used in this step to cover a broad range of use cases, or a single workload can be used to highly specialize the DBMS for that workload.

DEPLOYMENT PROCESS

First, the DBMS vendor identifies opportunities for field specialization. The vendor may already be aware of certain opportunities, or the vendor can use the Spiff Toolset to identify opportunities. The Spiff Toolset analyzes the DBMS both statically and dynamically. The dynamic analysis uses one or more representative workloads to rank candidate spiffs.

Next, spiffs are created based on the identified opportunities. The DBMS vendor can create the spiffs themselves, or the DBMS vendor can use the Spiff Toolset to create the spiffs.

Finally, the spiff-enhanced DBMS code is compiled by the DBMS vendor. The resulting DBMS binary is now enabled with field specialization. Once deployed, the DBMS will automatically specialize itself to each user's unique workload without any further action from either the DBMS vendor or the user.

In a typical use case, the DBMS vendor ranks candidate spiffs using the workload of the suite of Enterprise Application Software (EAS) applications that use their DBMS, such as CRM, ERP, BI, and HR applications. Since EAS applications will be bundled with a field-specialized DBMS, the vendor can market enhanced versions of each of those EAS applications, in addition to marketing field-specialized versions of their DBMS products.

Dataware's products can be easily incorporated into the vendor's existing software engineering processes and methodologies:

- Inserting spiffs results in small, local changes and retains the original semantics of the DBMS.
- As the DBMS code base evolves, existing spiffs can be updated by the DBMS vendor and new spiffs can be inserted to exploit any new invariants in the DBMS code.

COMPARISON WITH OTHER APPROACHES

Field specialization stands apart from existing approaches for improving DBMS performance due to its ease of incorporation into existing products, general applicability, and compatibility with other performance improvement approaches. See Table 2 for a tabular comparison.

Domain-Specific DBMS Architectures

Architectures specialized to particular domains, such as column-stores (e.g., MonetDB) and shared-nothing architectures (e.g., VoltDB), can offer significant performance improvements for databases that fall within their particular domain. Field specialization, in contrast, does not constrain the domain-applicability of a DBMS, nor does field specialization require rearchitecting the existing DBMS or applications built on top of the DBMS. In addition, field specialization can be applied to the aforementioned new architectures to further improve their efficiency.

Just-in-Time (JIT) Compilation

Field specialization is in some ways reminiscent of JIT compilation. While both involve runtime code optimization, their domains are very different. JIT compilation aims to speed up programs in managed languages such as Java and C# by optimizing away the interpretive overhead of byte code. Field specialization, by contrast, is applied to DBMSs implemented in C or C++ and optimizes the code by using information extracted from runtime invariants to specialize away low-level overheads.

Query Compilation

Query compilation exploits query-specific runtime information to compile a query into executable code, producing efficient code tailored to such runtime information. Nevertheless, performing compilation during query runtime incurs overhead that cannot be ignored, especially in OLTP workloads. In addition, maintaining a query compiler

	Field Specialization	Domain-Specific Architectures	JIT Compilation	Query Compilation
Improves legacy code	✓			
Domain independent	✓		✓	✓
Provides dynamic code optimization	✓		✓	✓
Applies to native code	✓	✓		✓
Avoids query-time compilation	✓	✓		
Effort to implement/maintain	Low	High	High	High

Table 2. *Approaches for DBMS vendors to improve the performance of their products*

on top of a standard-conformant DBMS, where new features are added frequently, is costly for the DBMS vendor. In contrast, field specialization is performed on the legacy DBMS code and takes advantage of runtime information without invoking the compiler at query runtime.

Field Specialization is Complementary

Table 2 on the previous page summarizes the pros and cons of these other approaches. We note however that field specialization is complementary to those approaches, and thus can be used in concert with any or all of them.

PERFORMANCE IMPACT

As previously mentioned, developers at Dataware have used field specialization to insert a modest number of spiffs into an industry-leading DBMS, PostgreSQL (PG) 9.3. Performance was compared using the TPC-H and TPC-DS benchmarks on hardware consisting of 4 x Intel Xeon E5-2670 v2 with 2 X 16GB (32GB) ECC DIMMs and a 800GB 9 GBPS SDD. The average speedups of the top five queries were 2.04X and 3.06X in the TPC-H and TPC-DS benchmarks, respectively. The average speedups across all queries in these benchmarks were 1.42X and 1.71X, respectively. Some of the benchmark queries did not exercise code that was specialized by these particular spiffs, presenting opportunities for additional spiffs that will increase the average speedups.

To see how the benefits of field specialization compare with those of current development efforts, two versions of PostgreSQL were compared on the TPC-H and TPC-DS benchmarks on the hardware described above, for the top five queries.

Comparison	TPC-H Speedup	TPC-DS Speedup	Development Effort
PG 9.1 vs. PG 9.3	1.17X	1.37X	130 man-months
PG 9.3 vs. field-specialized PG 9.3	2.04X	3.06X	2 man-months

Table 3. *Speedup vs. development cost*

As shown, field specialization allows better speedup with less effort. (Development from PG 9.1 to PG 9.3 wasn't fully focused on performance; these numbers should be interpreted accordingly.)

SUMMARY

DBMS vendors are expected to constantly improve the performance of their products. Field specialization is a patented new approach for improving DBMS performance that leverages runtime invariants in the DBMS, where the invariants arise from characteristics of the user's workload. These runtime invariants are used to create and insert spiffs into the DBMS; spiffs dynamically create specialized code at runtime, effectively allowing the DBMS to specialize itself at runtime and achieve significant runtime performance benefits.

Dataware offers an end-to-end solution to support the field specialization process: detecting runtime invariants, creating spiffs—simple code annotations around existing DBMS code—for those invariants, and managing spiffs and the specialized code they create at runtime.

Spiffs permit fine-grained control of the specialization process and customization to individual customer needs. Field specialization can be easily incorporated into legacy application code as well as into emerging technologies, such as cloud-based and main-memory DBMSs. Field specialization is complementary to other approaches to performance improvement and can work in concert with them, thereby yielding multiplicative speedups.

NEXT STEPS

To learn more about field specialization, visit www.datawareventures.com or email info@datawareventures.com.

DATAWARE VENTURES

9040 S. Rita Road, Suite 1270
Tucson, AZ 85747
+1 520 490 4843