ZipBurst™ Technical Backgrounder

For more information, contact David Dantowitz at 973-275-1024 or via email at david@dantowitz.com.

ZipBurst is a multi-threaded, MVCC, NoSQL resident database engine that accepts queries from Apache and other apps. Queries direct the engine to search for, select, and sort a set of result records. The resulting records are merged into a template file and returned as a MIME type determined within the template (XML, HTML, CSV, etc.) You can also generate a file to be post-processed, e.g., create a PHP file and return a URL to that file with a redirect so it is processed externally.

ZipBurst, written in C, includes a custom database search engine and two powerful interpreted programming languages (one for reports and one for mid-query computation). The app was written from scratch.

Languages

ZB-Tag Language (ZBTL) is a template or tag-based language that permits arbitrary embedding and is merged with data from the search engine to provide a query result. ZBTL is a complete and capable language. Using ZBTL enables customization of results and recursive calls to the database engine.

The second language is used primarily within the database engine for embedded computation but may also be used with ZBTL. This embedded language, akin to microcode is named ZipBurst Microcode Language, or ZBML. ZBML is a stack-based language with procedural support, early & late binding and simple function overloading. Using ZBML you can insert global and local computations at 10 points within a query.

For flexibility, ZBTL and ZBML can each embed the other's code. And, ZBML can be used to compute complex or dynamic sorting criteria for results.

Searching & Sorting

Queries include one or more directives to search for matching data. As you would expect, directives are composed of one or more values and a relationship, e.g.: equal, not equal, less than, greater than, contains, starts with, ends with, etc. Several data types are supported, including integers, floating point, dates, strings and so on. Once data is selected, sort directives indicate how to sort the results and they are passed on to be merged with a template file.

Query arguments, sort criteria, and other directives passed in via Apache may be static or dynamically computed using ZBML.

david@dantowitz.com
973-275-1024

Merging Results with a Template file

After a search is complete and the records sorted, the app reads a template file containing tags to be processed and merged with the data. A simple example might look like this:

```
<p>
[ZB-ResultCount] records were found.

Locations:
[ZB-Result]
        [ZB-Field name=City]<br>
[/ZB-Result]
```

Such template might yield

```
4 records were found.

Locations:
Boston
New York
Palo Alto
Seattle
```

ZipBurst processes a template file and interprets the “tags” which have the form:

[ZB-**tagname** …]

resulting in actions, text, or both.

The tags above are simple in nature:

1) Report how many results were found: **[ZB-ResultCount]**
2) Start and end a result sub-section, bounded with **[ZB-Result]** and **[/ZB-Result]**, note that the content within this pair of tags is repeatedly used for each resulting record found by the query (of course, you can cap the number records returned in a single query).
3) Retrieve specific data from the current matched record: **[ZB-Field name=fieldName]**

david@dantowitz.com
973-275-1024

More Advanced Tags

If/Then/Else
Loops, with the ability to jump to the start of the loop as well as exit (e.g., like continue and break in C)
Variables of type: string, int, float, date and boolean
Nesting of templates within templates (assists with modularization across an app)
Nested queries within a report template
Relational queries across multiple tables
Auto generation of links to next & previous pages of search results
An http call to another server

Nesting

Tags can also be nested arbitrarily, either in the body of a tag or a tag's argument. A simple example would nest a second If/Then/Else within the Then or Else body of an enclosing If/Then/Else.

A single If/Then/Else

[ZB-If arg1=… arg2=… op=…]
    True content
[ZB-Else]
    False content
[/ZB-If]

A nested If/Then/Else

[ZB-If arg1=… arg2=… op=…]
    True content
    [ZB-If arg1=… arg2=… op=…]
        Double True content
    [ZB-Else]
        True-False content
    [/ZB-If]
[ZB-Else]
    False content
[/ZB-If]

In addition to tags being nested, the value for any arguments within a tag may embed any tag:

[ZB-If arg1="**[ZB-Field name=city]**" arg2=… op=…]
    True content
[ZB-Else]
    False content
[/ZB-If]

david@dantowitz.com
973-275-1024

Nesting and Embedding are supported throughout the language. A tag (including an IF tag) may be embedded as an argument to a tag. And, just as an IF tag may included a nested IF tag, loops may be nested within loops. “Continue” and “Break” operate with respect to the scope of the current loop.

ZipBurst Microcode Language (ZBML)

If a query requires analytics, instrumentation, computation, complex or dynamic criteria, ZipBurst has a microcode-like language that is interpreted during the query and is able to use data in the database or data from external sources. You can compute and record information as data is processed during a query and use the results to augment searching, sorting and presentation. Mid-query computation can be used to create a computed column on the fly.

ZBML code can be executed at 10 distinct points within a query, providing great flexibility.

ZBML is a stack-based language similar to Forth or PostScript. It supports the data types: Int, Float, String, Lists, Literals and Procedures, stack-based operations, as well as early vs. late symbol binding, which can facilitate function overloading.

Other Details

Matching: ZipBurst also offers adjustable fuzzy-matching options when the data source and the query aren’t expected to be exact matches.

Web Form Argument Hiding: To simplify URLs and hide query details, you can specify static elements of an HTTP query in an argument file. This results in cleaner URLs and enables you to hide internal database details and lock query options.

Types of Data: Database files may be static or live.

Static data may be exported from other systems and used with ZipBurst exclusively for search and may not be modified.

Live data, is initially be imported from CSV or TAB delimited files (could embed JSON data as well). There is a multi-user web interface for accessing & editing data, with permissions-based operations. Live and static data are treated identically for query and report functions.

Multi-Version Concurrency Control: MVCC is an artifact of designing for sustained read and write operations with minimal latency. A few benefits arise from MVCC: if you begin a set of queries and the database is modified during those queries, you have the option of accessing the database as it was when you began. Additionally, with data spanning across time (subject to when garbage collection is done) you can instrument a query to use data from a past version rather than the current version.

david@dantowitz.com
973-275-1024