# ZipBurst™ Tech Backgrounder

For more information, contact David Dantowitz at 628-400-Code or via email at david@dantowitz.com.

ZipBurst is a multi-threaded NoSQL database engine that accepts queries from Apache and other apps. As expected, web-based queries direct a database search and a set of result records are selected. The resulting records are sorted using information from the query, and then merged into a template file. Based on the template file, the app can return any type of data (e.g., XML, HTML, CSV, etc.) You can even generate a file to be post-processed by a plug-in (e.g., a PHP file).

ZipBurst, written in C, includes a custom database search engine and two internal interpreted programming languages for developers to customize results.

## 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, able to perform any type of computation. Using ZBTL enables customization of results and even 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 and early & late binding. Using ZBML you can insert global and local computations at 10 points within a search query. ZBML was developed to address a client's request for daily changes within the database kernel as they fine-tuned search methods.

Of note is that ZBTL and ZBML can each embed the other's code.

## Searching & Sorting

Web queries include directives to search for data based on the expected sets of criteria across record elements (equal, not equal, less than, greater than, contains, starts with, ends, with etc.) and supports data types you would expect: integers, floating point, dates, strings and so on. Once data is selected based on the criteria, the results are then sorted, based on information from the web query.

Query arguments passed in via Apache can be static or dynamically computed using ZBML.

## 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 the 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]**, noting that the content within is repeated for each resulting record found by the query (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
628-400-Code

## 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)
Variable Support for strings, ints, floats, dates, and booleans
Nested queries within the report template
Relational queries across multiple tables / files
An http call to another server (e.g., get the current temperature)
and others …

## Nesting

Tags can also be nested arbitrarily. The simplest 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]

Even an IF tag could be used within an argument. Loops, too, may be nested.


david@dantowitz.com
628-400-Code

## ZipBurst Microcode Language (ZBML)

If the query you have in mind requires more complex or dynamic criteria, ZipBurst contains a microcode-like language that is interpreted during the query. This akin to creating a computed column, but with the ability to perform arbitrary computation or record data along the way.

ZBML is be executed at 10 distinct points within a query, enabling much flexibility. You can store static ZBML on the server or include it with a query.

ZBML is a stack-based language similar to Forth and PostScript. It supports the data types: Int, Float, String, Lists, Literals and Procedures, stack-based operations, as well as an option for early vs. late binding.

## Other Details

**Matching:** ZipBurst also offers adaptive matching / spelling options for queries where the data source and the query aren't expected to be exact matches.

**Web Form Argument Hiding:** In order to simply URLs and hide details of queries, you can store static elements of an HTTP query in an argument file. This results in cleaner URLs and enables you to hide some internal database details.

**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. CSV or TAB delimited files (could embed JSON data as well) are supported. Live data is supported with a multi-user web interface for accessing & editing data. The live and static data are treated identically by the database engine.

**Data Time-Spanning:** DTS is an artifact of designing for sustained read and write operations with minimal latency. A few benefits arise from DTS: if you begin a set of queries and the database is changed during those queries, you can continue accessing the database as it was when you started. Obviously, there are situations when it would be handy to access a database at time slices other than the most current.


david@dantowitz.com
628-400-Code