Google Summer of Code 2019 Proposal - PostgreSQL

TOAST'ing in slices

Personal Information

Name: Bruno Hass

Country: Brazil

Email: bruno_hass@live.com

Phone: 5548984704445

List of Deliverables

- Modify JSONB TOASTs to include data about keys contained in the chunk.
- Support for JSONB queries capable of deTOAST'ing only the relevant chunks to the query, e.g. chunks containing the desired JSON keys.
- Modify array TOASTs to include information about the range of elements present in the chunk.
- Support for array queries capable of deTOAST'ing only the relevant chunk containing the element or range of elements queried.
- Modify text TOASTs to be split into characters boundaries, not bytes.
- Improve text pattern matching by gradually deTOAST'ing chunks and pattern match them.

Description

This proposal aims to modify how some types of TOASTed values are queried and stored based on the type of the data. Currently, PostgreSQL splits values that exceed one page in size into compressed chunks of 2KB and stores them in a separate table. The drawback of the current approach is that it might become quite slow for queries for the TOASTed data. Consider, for instance, a table with a JSONB column containing several rows with TOASTed values for this column. If we query a particular JSON key we must decompress all the TOASTed JSONB chunks in order to check for the key value. The changes of this proposal will modify the TOAST table row to contain information about the stored value in order to make queries faster. For this example, each TOAST table entry would contain information about which JSON keys that chunk has, so we need only to decompress the relevant chunks. The same principle might be applied to other variable length data types. For arrays we could keep information on the TOAST table entry about which range of elements that chunk contains. For text fields we could split them into characters boundaries instead of byte boundaries, allowing faster searches for substring patterns.

Approach Outline

The first step necessary to support the optimized functionalities from the description is to modify how the TOAST table works. Some kind of metadata must be added to the TOAST table entry. The content of such metadata field will depend on the type of the data. My approach would be to add a new column to the TOAST table to contain such metadata. The next step would be to modify how the functions responsible for TOAST'ing JSONBs, arrays and text works. When TOAST'ing a JSONB we shall split it into chunks while keeping the tree structure of the JSONB intact inside a chunk , except for possibly missing childs of nodes, which are present in other chunks. The TOAST metadata field of each chunk will store information about which keys are present in the chunk. The function responsible for accessing a field from a TOASTed JSONB would read the metadata field and skip the TOAST rows not containing the desired key.
For TOAST'ing an array we should split it into slices and store each slice in a chunk. The metadata field would inform the range of the slice contained in the chunk. When accessing an array element the responsible function would skip the chunks not containing the desired element. Finally, for text we are going to modify the TOAST'ing function to split at characters boundaries, so substring pattern matching can be modified to gradually decompress the rows of the TOASTed value and to apply the comparison to the chunk, stopping if it finds the pattern.