

Method of encoding Tag/Length/Value for partial search

Description

Method of Encoding and decoding tag/length/value (TLV) key-information for partial match when searching for records.

Technical Field

Tag length value (TLV) is a method of encoding structured information in an opaque buffer form by computer systems that can be decoded to recreate the original structured information later.

The tag identifies the information that follows, the length identifies the number of bytes following that constitute the value, while the value identifies the content encoded as several bytes.

The content can be primitive values that can be decoded directly or structured as nested tag/length/value. By encoding parts as TLV enables complex information to be encoded and decoded.

Applications that use TLV need to know the meaning of each tag to consistently encode and decode information.

Background

When the full key is known, the encoded opaque buffer can be used to search for the unique matches.

[100] Represents five Names that have been encoded using TLV using the record Name with a tag of '1'. Within Name there are two parts encoded with tag 2 (Surname) and tag 3 (first name).

The unique name "Doe, Jon" can be found by TLV encoding the name and searching for a match. TLV encoding cannot be used to search for any person whose surname is "Doe" since

the ordering of TLV buffers will place "Doe, Ade" near the top of a list and "Doe, Zachariah" near the bottom since the length of the name is the second part in the opaque buffer.

TLV can be used to find a unique match, but the full list must be searched if only part of the key is known.

Standard Approach

The standard approach is to extract the key information (e.g. using a domain delimiter) and only use the key for indexing and store all information (key and value) as content.

Claim

1. Reordering the bytes in an opaque buffer from Tag/Length/Value to Tag/Value/Length (TVL) results in a list that places records that start with the same value together but retains the ability to transparently decode the information [100] shows an ordered list of five names.

Surname	First Name
Cox	Jon
Doe	Jon
Cox	John
Doe	John
Smith	Jo

and [101] shows list once encoded with Tag/Value/Length.

Surname	First Name
Cox	John
Cox	Jon
Doe	John
Doe	Jon
Smith	Jo

2. Applying the TLV to TVL encoding to a search template enables a range search of values without reference to the domain structure of information.

[103] Show a TLV of record that only

contains surname, and [104] once TVL encoded.

Replacing the length values with '0' ('\0' binary byte value) for start of range and high values for end of range ('ZZ' in the diagram, but '0xFF' binary byte value) creates a start/end pair for search keys that are within the range required [105], since the next tag ('2' in the example) will always be higher than '\0' and lower than '0xFF' regardless of the actual content. [106] shows an example of the range matching.

Application

Tag/Value/Length encoding enables key-store databases to be searched with a single indexed scan for keys that match the domain intent without storing separate indexes for partial matches or scanning all values.

When the table of keys are distributed over several devices each scan can be performed in parallel.

Encode Method

The method relies on basic information of which tag values represent nested TLV buffers (1: (2,3)) in the example, but not domain knowledge.

Create an output buffer of the same length as the source and note:

- S the start position of the buffer = 0
- E the end position of the buffer = end
- P the current position in the input buffer = 0

Starting with the first byte in the TLV buffer

1. Copy the tag at P from input to S output. increment P and S.
2. If the length is implied by the tag (e.g. 32-bit number), copy the value from P to S. increment P and S.

3. If the tag is followed by a length copy the value from P to E. increment P and decrement E.
4. If the value is not a nested TLV buffer, copy the value from P to S for the length of the value. Increment P and S.

Repeat until P has reached the end of the input buffer. [107] shows an example input buffer and [108] the construction of the output buffer.

Decode Method

Create an output buffer of the same length as the source and note:

- S the start position of the buffer = 0
- E the end position of the buffer = end
- P the current position in the input buffer = 0

Starting with the first byte in the TLV buffer

5. Copy the tag at P from input to S output. increment P and S.
6. If the length is implied by the tag (e.g. 32-bit number), copy the value from P to S. increment P and S.
7. If the tag has a length copy the value from E in input to S in output. increment S and decrement E.
8. If the value is not a nested TLV buffer, copy the value from P to S for the length of the value. Increment P and S.

Repeat until P is equal to E.

Encode TVL buffer for ranger search.

Perform the same procedure for decoding but replacing each length part with '\0' for start and 0xFF for end.

