**Overview**

**This script implements a highly optimized HBase table structure for storing and querying web page content with associated metadata and link relationships. The design addresses common web content management requirements while ensuring high performance at scale.**

**Table Design Rationale**

**Column Family Structure**

**We designed four column families to optimize different access patterns:**

**content**

* Versions: 3 (stores up to 3 versions of page content)
* TTL: 90 days (7776000 seconds)
* Compression: SNAPPY (reduces storage by 50%)
* Bloom Filter: ROWCOL (optimizes for both row and column access)
* IN\_MEMORY: true (prioritizes caching for frequent reads)

**metadata**

* Versions: 1 (no versioning needed for basic metadata)
* No TTL (persists indefinitely)
* ROW bloom filter (optimized for row-key lookups)
* outlinks and inlinks
* Versions: 2 (tracks link changes)
* TTL: 180 days (15552000 seconds)
* Compression: SNAPPY (link data can be large)

**Row Key Design:**

**The composite row key structure was carefully engineered for optimal performance:**

<reverse\_domain>|<url\_hash>|<reverse\_timestamp>

**Components:**

* Reverse Domain (e.g., com.example instead of example.com)
* Groups all pages from the same domain together physically
* Enables efficient range scans for domain-specific queries
* Example: All example.com pages are stored contiguously
* URL Hash (First 8 chars of SHA-1)
* Prevents hotspotting from sequential URLs
* Maintains deterministic lookups while ensuring even distribution
* Example: a1b2c3d4 for http://example.com/page1
* Reverse Timestamp (MAX\_LONG - actual\_timestamp)
* Automatically sorts newest content first
* Enables efficient time-range queries without client-side sorting
* Example: 9223372036854775807 - 1672531200000

**Implementation Details :**

**Data creation :**

**# Create table with 4 column families and optimized settings**

**create 'webtable',**

**{NAME => 'content', VERSIONS => 3, TTL => 7776000, COMPRESSION => 'SNAPPY', BLOOMFILTER => 'ROWCOL', IN\_MEMORY => true},**

**{NAME => 'metadata', VERSIONS => 1, COMPRESSION => 'SNAPPY', BLOOMFILTER => 'ROW'},**

**{NAME => 'outlinks', VERSIONS => 2, TTL => 15552000, COMPRESSION => 'SNAPPY'},**

**{NAME => 'inlinks', VERSIONS => 2, TTL => 15552000, COMPRESSION => 'SNAPPY'}**

**# Pre-split table at major domain boundaries**

**split 'webtable', ['com.example|', 'org.test|', 'net.sample|', 'edu.univ|', 'gov.data|']**

**Data Generation :**

**The Python script generates realistic sample data (on github)**

**Implementing business requirements :**

**# Content Management Operations**

**echo "1. Retrieve latest version of example.com/page1:"**

**get 'webtable', 'com.example|a1b2c3d4|9223372036854775807-1672531200000'**

**echo "2. List pages from example.com (first 5):"**

**scan 'webtable', {STARTROW => 'com.example|', STOPROW => 'com.example|\xFF', LIMIT => 5}**

**# SEO Analysis Operations**

**echo "3. Find pages linking to example.com/page2:"**

**scan 'webtable', {COLUMN => 'inlinks:http://example.com/page2', FILTER => "ValueFilter(=, 'binary:1')"}**

**echo "4. Identify pages with no outbound links:"**

**scan 'webtable', {COLUMN => 'outlinks:', FILTER => "ColumnCountGetFilter(1) AND ValueFilter(=, 'binary:0')"}**

**Performance Considerations**

* Pre-Splitting
* Bash
* split 'webtable', ['com.example|', 'org.test|', 'net.sample|', 'edu.univ|', 'gov.data|']
* Prevents initial hotspotting during data loading
* Aligns with domain boundaries for efficient scans
* Creates balanced regions from the start
* Compression
* SNAPPY compression reduces:
* Storage requirements by ~50%
* Network transfer times
* Disk I/O during scans
* Minimal CPU overhead (2-5% performance impact)
* Bloom Filters
* ROWCOL for content family (frequent column-specific access)
* ROW for metadata (mostly accessed by row key)
* Reduces disk seek