**Overview**

The system is designed to store webpage content, metadata, and link relationships, with comprehensive query capabilities for SEO analysis and performance monitoring.

System Architecture

Data Model

Table Name:

webtable

Column Families:

content: Stores HTML content with versioning (3 versions retained)

metadata: Stores page metadata (title, status code, creation date, size)

outlinks: Stores external links from each page

inlinks: Stores incoming links to each page (inverted index)

Key Design Features

Row Key: URL of the webpage (e.g., "http://example.com/page1")

TTL Settings:

Content: 90 days (7,776,000 seconds)

Links: 180 days (15,552,000 seconds)

Versioning:

Content: 3 versions maintained

Links: 2 versions maintained

Data Generation

Python Data Generator

The system includes a Python script that generates realistic web page data using the Faker library you will find in the repo named as populate\_webtable.py

Core Operations

Basic CRUD Operations

Insert Data:

hbase

put 'webtable', 'http://example.com/page1', 'content:html', '<html>...'

put 'webtable', 'http://example.com/page1', 'metadata:title', 'Example Page'

Retrieve Data:

hbase

get 'webtable', 'http://example.com/page1'

Update Content:

hbase

put 'webtable', 'http://example.com/page1', 'content:html', 'Updated content'

Delete Data:

hbase

deleteall 'webtable', 'http://example.com/page1'

Version Control

hbase

get 'webtable', 'http://example.com/page1', {COLUMN => 'content:html', VERSIONS => 3}

Advanced Query Capabilities

Domain-Specific Queries

hbase

scan 'webtable', {STARTROW => 'http://example.com', ENDROW => 'http://example.com~'}

Keyword search

hbase

scan 'webtable', {

FILTER => "ValueFilter(=, 'substring:contact us')", COLUMNS => 'content:html'

}

Title search:

hbase

scan 'webtable', {

FILTER => "ValueFilter(=, 'substring:privacy')", COLUMNS => 'metadata:title'

}

Performance Optimization Queries

Size-Based Analysis

hbase

scan 'webtable', {

FILTER => "SingleColumnValueFilter('metadata', 'size', >=, 'binary:4000')"

}

Pagination

hbase

scan 'webtable', {LIMIT => 5}

hbase

scan 'webtable', {STARTROW => 'http://test.org/page5', LIMIT => 5}

Row Key Design & Pagination Strategy

1. Row Key Structure

The row keys are designed as complete URLs (e.g., http://example.com/page1) to enable:

Natural grouping – Pages from the same domain are stored contiguously.

Efficient scans – Range queries can fetch all pages under a domain.

2. Pagination Mechanism

Since HBase does not natively support offset-based pagination, we use row key tracking:

Initial request: Fetch the first N records (LIMIT).

Subsequent requests: Use the last seen row key as STARTROW to get the next batch.

Domain-specific pagination: Narrow scans using STARTROW and ENDROW to paginate within a domain.

3. Performance Considerations

Avoid hotspots: URL-based keys distribute writes evenly.

Optimal batch size: Balance between too many small scans (inefficient) and large memory-heavy batches.

Client-side tracking: Maintain the last row key to resume pagination.

This design ensures scalable, sorted access while leveraging HBase’s natural key-ordering for efficiency.