HBase WebTable Case Study - Business Requirements

Background

Your company manages a web content system that needs to store and efficiently retrieve web pages, their metadata, and link structures. You'll implement this using HBase as the backend storage system.

System Requirements

1. Design and implement an HBase table to store web page data
2. Implement data ingestion for sample web pages
3. Create queries to support various business access patterns
4. Design appropriate time-to-live (TTL) and versioning policies
5. Implement filtering and pagination mechanisms for efficient data retrieval

Part 1: Table Design & Implementation

Task 1.1: Create the HBase Table

Create an HBase table with appropriate column families, versioning, and TTL settings:

- Content family: Store HTML content with 3 versions and 90-day TTL
- Metadata family: Store page metadata with 1 version and no TTL
- Outlinks family: Store outbound links with 2 versions and 180-day TTL
- Inlinks family: Store inbound links with 2 versions and 180-day TTL

Task 1.2: Data Generation

Generate and insert at least 20 sample web pages with the following characteristics:

- 5 different domains (e.g., example.com, test.org)
- Various content sizes (small, medium, large pages)
- Different creation dates (some recent, some older)
- Interconnected link structure (pages linking to each other)

Part 2: Business Access Patterns

Business Requirement 1: Content Management

The content team needs to:

- Retrieve the latest version of any page by URL
- View historical versions of a page to track changes
- List all pages from a specific domain for content audits
- Find all pages modified within a specific time range

Business Requirement 2: SEO Analysis

The SEO team needs to:

- Find all pages linking to a specific URL (inbound links)
- Identify pages with no outbound links (dead ends)
- List pages with the most inbound links (popular pages)
- Retrieve pages with specific content in the title or body

Business Requirement 3: Performance Optimization

The performance team needs to:

- Identify the largest pages by content size
- Find pages with HTTP error status codes
- List pages with outdated content (not modified in last 30 days)

Part 3: Implementation Tasks

Task 3.1: Basic Operations

Implement HBase shell commands to:

- Insert complete web page data (content, metadata, links)
- Retrieve a page by exact URL
- Update a page's content and metadata
- Delete a page and all its information

Task 3.2: Filtering Operations

Implement HBase shell commands with filters to:

- Find pages with titles containing specific keywords
- Retrieve pages with content size above a threshold
- List pages with specific HTTP status codes
- Find pages modified after a specific date

Task 3.3: Scanning with Pagination

Implement pagination mechanisms to:

- Scan domain pages in batches of 5 records
- Retrieve large result sets efficiently
- Implement "next page" functionality using row key markers
- Demonstrate how pagination improves query performance

Task 3.4: Time-Based Operations

Implement operations that leverage versioning and TTL:

- Compare different versions of the same page
- Demonstrate how TTL automatically removes old content
- Implement a manual purge for outdated content
- Show how to retrieve the latest N versions of content

Deliverables

1. HBase shell script for table creation with proper settings
2. Data generation script using python Faker library.
3. Query scripts for each business requirement
4. Documentation explaining the design decisions including:
    - Row key design rationale
    - Versioning and TTL policy justifications
    - Filter selection for different query types
    - Pagination implementation details

Evaluation Criteria

- Correct implementation of table design with appropriate settings
- Effective row key design for load balancing and scanning
- Proper use of HBase filters and scan operations

- Efficient pagination implementation, Clear documentation of design decisions and tradeoffs