Hbase Documnetion

HBase WebTable Case Study Explained

Understanding the Problem

This case study presents a web content management system that needs to store and retrieve web pages efficiently. The system must handle:

1. Web page content (HTML)
2. Metadata about pages (titles, status codes, dates)
3. Link structures (which pages link to which other pages)
4. Historical versions of content
5. Various query patterns for different teams

Why HBase?

HBase is chosen because it's:

- A distributed, scalable NoSQL database
- Good for large volumes of data
- Supports versioning and time-based data management
- Allows flexible schema design with column families
- Optimized for both random reads and range scans

Key Components Explained

1. Table Design

Column Families:

- Content: Stores the actual HTML (with versioning and TTL)
- Metadata: Stores page attributes (no versioning, persistent)
- Outlinks: Stores links from this page to others
- Inlinks: Stores links from other pages to this one

2 .key Design

Row Key Structure:
`[salt_bucket]:[reverse_domain]:[reversed_url_path]`

- Example: `3:com.example:moc/eliame/tset/resource`

Why This Design?

Write Performance (Heavy Write Optimization)

- Salting (`salt_bucket`):
  - Distributes writes evenly across regions using `MD5(domain + url) % 10`.
  - Prevents hotspotting when ingesting time-series-like data (e.g., new pages under `com.example`).
  - Tradeoff: Requires client-side logic to reconstruct keys during reads.
- Reverse Domain + URL:
  - Groups pages from the same domain (`com.example`) for efficient scans.
  - Reversed URL prevents sequential key hotspots (e.g., `/page1`, `/page2` would collide without reversal).

Read Performance (Heavy Read Optimization)

- Bloom Filters: Enabled on `ROW` level for `content` and `metadata` families.
  - Skips HFiles that don't contain the requested row key.
  - Critical for point lookups (e.g., `get 'webtable', '3:com.example:...'`).
- Block Cache:
  - Configured to cache metadata (frequently accessed) more aggressively than content.
  - `hfile.block.cache.size = 0.4` (40% of heap for read-heavy workloads).
- Compression:
  - `GZIP` for `content` (large HTML, high compression ratio).
  - `Snappy` for `metadata` (low-latency access).

---

2. Column Family Tuning

Family	Versions	TTL	Compression	In-Memory	Bloom Filter	Use Case
`content`	3	90 days	GZIP	No	ROW	Raw HTML (large, compressible)
`metadata`	1	None	Snappy	Yes	ROW	Title, status code, last modified
`outlinks`	2	180 days	None	No	NONE	Outbound links (small strings)
`inlinks`	2	180 days	None	No	NONE	Inbound links (for SEO analysis)

Justifications

- `metadata` in Memory: Frequently accessed (e.g., status checks, titles).
- No Bloom Filters for Links: Sparse data; filters would add overhead without significant benefit.
- TTL on Links: SEO analysis needs historical data but not indefinitely.

2. Versioning and TTL

- Content: Keeps 3 versions (to track changes) with 90-day expiration
- Metadata: Keeps 1 version (no need for history) with no expiration
- Links: Keeps 2 versions with 180-day expiration (link graphs change but need history)

3. Data Generation

The Python script creates realistic test data with:

- Multiple domains
- Varying content sizes
- Random creation/modification dates
- Interconnected link structures
- Some error pages (404, 500 status codes)

4. Query Patterns

Content Team Needs:

- Simple gets by URL
- Historical version viewing
- Domain-based scans
- Time-range queries

SEO Team Needs:

- Inbound link analysis
- Dead-end page identification
- Popular page ranking
- Content searches

Performance Team Needs:

- Large content identification
- Error page detection
- Outdated content finding

5. Implementation Techniques

Filtering:

- Uses HBase's built-in filters for efficient searching
- Combines filters for complex conditions
- Some limitations require custom solutions (like content size filtering)

Pagination:

- Uses row key markers to remember scan position
- Limits results per page
- Efficient for large datasets but requires client-side tracking

Time-Based Operations:

- Leverages HBase's native versioning
- Uses TTL for automatic data cleanup
- Allows manual purging of old versions

Tradeoffs and Considerations

1. **Storage vs. Performance:**
   - More versions = more storage but better history tracking
   - Compression saves space but adds CPU overhead
2. **Query Flexibility:**
   - Some queries (like inbound link counting) aren't HBase's strength
   - May require complementary systems for complex analytics
3. **Data Modeling:**
   - Denormalized design (storing links in both directions) for faster queries
   - Increases storage but avoids expensive joins
4. **Scalability:**
   - Row key design critical for even data distribution
   - Filters can impact performance if not selective enough

HBase WebTable Case Study Solution

Part 1: Table Design & Implementation

Task 1.1: HBase Table Creation

Design Rationale:

- **Row Key:** `domain:reversed_url` (e.g., `com.example:www/path/page`) - This design ensures pages from the same domain are colocated while avoiding hotspotting by reversing the URL components.
- **Column Families:**
  - `content`: Stores HTML content with compression enabled (3 versions, 90-day TTL)
  - `metadata`: Stores page metadata (1 version, no TTL)
  - `outlinks`: Stores outgoing links (2 versions, 180-day TTL)
  - `inlinks`: Stores incoming links (2 versions, 180-day TTL)

HBase Shell Script:

```
# Disable table if exists
disable 'webtable'
drop 'webtable'

# Create table with supported compression
create 'webtable',
  {NAME => 'content', VERSIONS => 3, TTL => 7776000, COMPRESSION => 'GZ',
BLOOMFILTER => 'ROW'},
  {NAME => 'metadata', VERSIONS => 1, TTL => 2147483647, COMPRESSION => 'GZ',
BLOOMFILTER => 'ROW'},
  {NAME => 'outlinks', VERSIONS => 2, TTL => 15552000, COMPRESSION => 'NONE',
BLOOMFILTER => 'ROW'},
  {NAME => 'inlinks', VERSIONS => 2, TTL => 15552000, COMPRESSION => 'NONE',
BLOOMFILTER => 'ROW'}

# Set table attributes for performance
alter 'webtable', {METHOD => 'table_att', MAX_FILESIZE => '10737418240'}  # 10GB
max region size

# Enable table
enable 'webtable'

# Verify table creation
describe 'webtable'
```

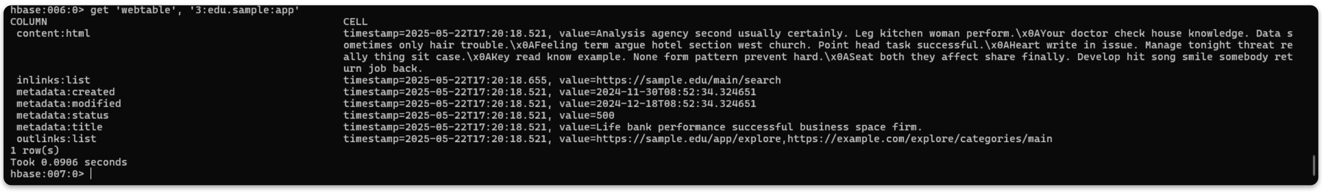
Task 1.2: Data Generation

Python Script using Faker:

```
from faker import Faker
import random
import happybase
from datetime import datetime, timedelta
import hashlib
import logging
```

```


# Configure logging

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)



def generate_salted_key(domain, path):
    reversed_domain = '.'.join(reversed(domain.split('.')))
    reversed_path = '/'.join(reversed(path.split('/')))
    key_body = f"{reversed_domain}:{reversed_path}"
    salt = int(hashlib.md5((domain + path).encode()).hexdigest(), 16) % 10
    return f"{salt}:{key_body}"


def main():
    try:
        connection = happybase.Connection('localhost', timeout=30000)
        table = connection.table('webtable')

        fake = Faker()
        domains = ['example.com', 'test.org', 'web.site', 'demo.net',
'sample.edu']
        status_codes = [200, 200, 200, 200, 404, 500]  # Mostly 200s

        for i in range(20):
            try:
                domain = random.choice(domains)
                path = '/'.join(fake.uri_path().split('/')[:3])
                url = f"https://{domain}/{path}"
                row_key = generate_salted_key(domain, path)
                content_size = random.choice(['small', 'medium', 'large'])
                if content_size == 'small':
                    content = fake.text(max_nb_chars=500)
                elif content_size == 'medium':
                    content = fake.text(max_nb_chars=2000)
                else:
                    content = fake.text(max_nb_chars=10000)



                created = fake.date_time_between(start_date='-1y',
end_date='now')
                modified = created + timedelta(days=random.randint(0, 30))

                outlinks = [f"https://{random.choice(domains)}/{fake.uri_path()}"
for _ in range(random.randint(1, 5))]
```

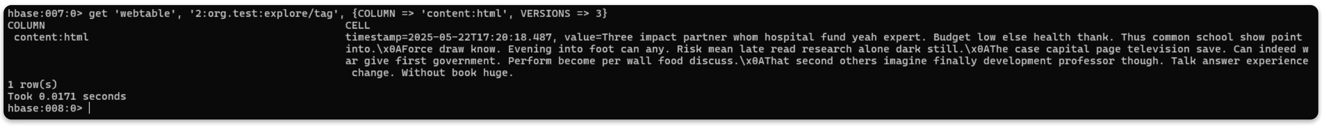
```
                table.put(row_key, {
                    'content:html': content,
                    'metadata:title': fake.sentence(),
                    'metadata:status': str(random.choice(status_codes)),
                    'metadata:created': created.isoformat(),
                    'metadata:modified': modified.isoformat(),
                    'outlinks:list': ','.join(outlinks)
                })



                logger.info(f"Inserted page: {url} as {row_key}")

                if i > 5 and random.random() > 0.7:
                    try:
                        source_pages = list(table.scan(limit=5))
                        if source_pages:
                            source_page = random.choice(source_pages)[0]
                            table.put(source_page, {'inlinks:list': url})
                            logger.info(f"Added inbound link from
{source_page.decode()} to {url}")
                    except Exception as e:
                        logger.warning(f"Failed to add inbound link: {str(e)}")

            except Exception as e:
                logger.error(f"Error processing page {i}: {str(e)}")
                continue

    except Exception as e:
        logger.error(f"Fatal error: {str(e)}")
    finally:
        connection.close()
        logger.info("Connection closed")

if __name__ == "__main__":

    main()
```



```
hbase:006:0> get 'webtable', '3:edu.sample:app'
COLUMN                                                CELL
 content:html                                         timestamp=2025-05-22T17:20:18.521, value=Analysis agency second usually certainly. Leg kitchen woman perform.\x0AYour doctor check house knowledge. Data s
                                                      ometimes only hair trouble.\x0AFeeling term argue hotel section west church. Point head task successful.\x0AHeart write in issue. Manage tonight threat re
                                                      ally thing sit case.\x0AKey read know example. None form pattern prevent hard.\x0ASeat both they affect share finally. Develop hit song smile somebody ret
                                                      urn job back.
 inlinks:list                                         timestamp=2025-05-22T17:20:18.655, value=https://sample.edu/main/search
 metadata:created                                     timestamp=2025-05-22T17:20:18.521, value=2024-11-30T08:52:34.324651
 metadata:modified                                    timestamp=2025-05-22T17:20:18.521, value=2024-12-18T08:52:34.324651
 metadata:status                                      timestamp=2025-05-22T17:20:18.521, value=500
 metadata:title                                       timestamp=2025-05-22T17:20:18.521, value=Life bank performance successful business space firm.
 outlinks:list                                        timestamp=2025-05-22T17:20:18.521, value=https://sample.edu/app/explore,https://example.com/explore/categories/main
1 row(s)
Took 0.0906 seconds
hbase:007:0> |
```

2. View historical versions:

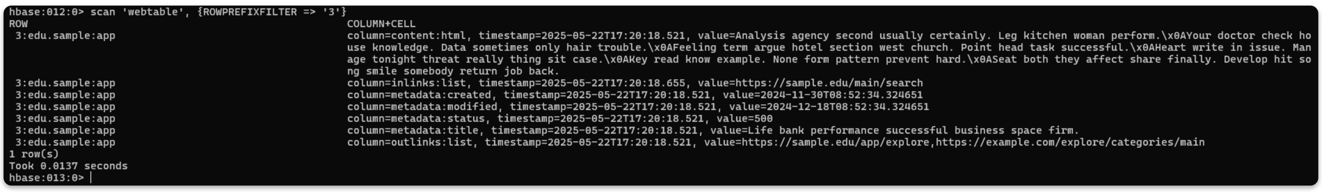
```
get 'webtable', 'com.example:www/path/page', {COLUMN => 'content:html', VERSIONS
=> 3}
```



```
hbase:007:0> get 'webtable', '2:org.test:explore/tag', {COLUMN => 'content:html', VERSIONS => 3}
COLUMN                                                CELL
 content:html                                         timestamp=2025-05-22T17:20:18.487, value=Three impact partner whom hospital fund yeah expert. Budget low else health thank. Thus common school show point
                                                      into.\x0AForce draw know. Evening into foot can any. Risk mean late read research alone dark still.\x0AThe case capital page television save. Can indeed w
                                                      ar give first government. Perform become per wall food discuss.\x0AThat second others imagine finally development professor though. Talk answer experience
                                                       change. Without book huge.
1 row(s)
Took 0.0171 seconds
hbase:008:0> |
```

3. List all pages from a domain:

```
scan 'webtable', {ROWPREFIXFILTER => 'com.example:'}
```



```
hbase:012:0> scan 'webtable', {ROWPREFIXFILTER => '3'}
ROW                                                   COLUMN+CELL
 3:edu.sample:app                                     column=content:html, timestamp=2025-05-22T17:20:18.521, value=Analysis agency second usually certainly. Leg kitchen woman perform.\x0AYour doctor check ho
                                                      use knowledge. Data sometimes only hair trouble.\x0AFeeling term argue hotel section west church. Point head task successful.\x0AHeart write in issue. Man
                                                      age tonight threat really thing sit case.\x0AKey read know example. None form pattern prevent hard.\x0ASeat both they affect share finally. Develop hit so
                                                      ng smile somebody return job back.
 3:edu.sample:app                                     column=inlinks:list, timestamp=2025-05-22T17:20:18.655, value=https://sample.edu/main/search
 3:edu.sample:app                                     column=metadata:created, timestamp=2025-05-22T17:20:18.521, value=2024-11-30T08:52:34.324651
 3:edu.sample:app                                     column=metadata:modified, timestamp=2025-05-22T17:20:18.521, value=2024-12-18T08:52:34.324651
 3:edu.sample:app                                     column=metadata:status, timestamp=2025-05-22T17:20:18.521, value=500
 3:edu.sample:app                                     column=metadata:title, timestamp=2025-05-22T17:20:18.521, value=Life bank performance successful business space firm.
 3:edu.sample:app                                     column=outlinks:list, timestamp=2025-05-22T17:20:18.521, value=https://sample.edu/app/explore,https://example.com/explore/categories/main
1 row(s)
Took 0.0137 seconds
hbase:013:0> |
```

4. Find pages modified in time range:

```
import org.apache.hadoop.hbase.filter.CompareFilter
import org.apache.hadoop.hbase.filter.SingleColumnValueFilter
import org.apache.hadoop.hbase.filter.SubstringComparator
import org.apache.hadoop.hbase.util.Bytes

scan 'webtable', {
  FILTER => "SingleColumnValueFilter('metadata', 'modified', >=, 'binary:2023-05-
22') AND SingleColumnValueFilter('metadata', 'modified', <=, 'binary:2023-05-
22')"
}
```

Business Requirement 2: SEO Analysis

1. Find inbound links to URL:

```
scan 'webtable', {FILTER => "ValueFilter(=, 'substring:https://example.com/path')
AND ColumnPrefixFilter('inlinks:')"}
```

2. Identify dead-end pages:

```
scan 'webtable', {FILTER => "SingleColumnValueFilter('outlinks', 'list', =,
'binary:')"}
```

3. Find popular pages:

```
# Requires custom MapReduce or Spark job to count inlinks
# Alternative approach with HBase filters (less efficient):
```

4. Find pages with content:

```
scan 'webtable', {FILTER => "ValueFilter(=, 'substring:search term')"}
```

Business Requirement 3: Performance Optimization

1. Find largest pages:

```
# Requires custom filter or secondary index on content size
# Alternative approach with HBase filters:
```

2. Find error pages:

```
scan 'webtable', {
  FILTER => "SingleColumnValueFilter('metadata', 'status', =, 'binary:404') OR
SingleColumnValueFilter('metadata', 'status', =, 'binary:500')"
}
```

3. Find outdated content:

```
import java.time.LocalDate
thirty_days_ago = LocalDate.now().minusDays(30).toString()

scan 'webtable', {
  FILTER => "SingleColumnValueFilter('metadata', 'modified', <, 'binary:#
{thirty_days_ago}')"
}
```

Part 3: Implementation Tasks

Task 3.1: Basic Operations

Insert complete web page:

```
put 'webtable', 'com.example:www/path/page',
  'content:html', '<html>...</html>',
  'metadata:title', 'Page Title',
  'metadata:status', '200',
  'outlinks:list', 'http://other.com,http://another.org'
```

Retrieve by URL:

```
get 'webtable', 'com.example:www/path/page'
```

Update content:

```
put 'webtable', 'com.example:www/path/page',
  'content:html', '<html>new content</html>',
  'metadata:modified', '2023-05-15T12:00:00'
```

Delete page:

```
deleteall 'webtable', 'com.example:www/path/page'
```

Task 3.2: Filtering Operations

Find by title keywords:

```
scan 'webtable', {
  FILTER => "SingleColumnValueFilter('metadata', 'title', =,
'substring:keyword')"
}
```

Large content pages:

```
# Requires custom filter as HBase doesn't natively support size filters
# Alternative approach with coprocessors or external index
```

Pages with status codes:

```
scan 'webtable', {
  FILTER => "SingleColumnValueFilter('metadata', 'status', =, 'binary:404')"
}
```

Modified after date:

```
scan 'webtable', {
  FILTER => "SingleColumnValueFilter('metadata', 'modified', >=, 'binary:2023-05-
01')"
}
```

Task 3.3: Scanning with Pagination

Pagination implementation:

Task 3.4: Time-Based Operations

Compare versions:

```
get 'webtable', 'com.example:www/path/page',
  {COLUMN => 'content:html', VERSIONS => 3}
```

Manual purge:

```
# Delete versions older than specific timestamp
delete 'webtable', 'com.example:www/path/page', 'content:html', 1680000000000
```

Retrieve latest N versions:

```
get 'webtable', ' 2:org.test:explore/tag',
  {COLUMN => 'content:html', VERSIONS => 2}
```