hbase webpages schema design

Task 1.1: Create the HBase Table

- 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

```
create 'webpages',
  {NAME => 'content', VERSIONS => 3, TTL => 7776000},
  {NAME => 'metadata', VERSIONS => 1},
  {NAME => 'outlinks', VERSIONS => 2, TTL => 15552000},
  {NAME => 'inlinks', VERSIONS => 2, TTL => 15552000}
```

Task 1.2: Data Generation

- python script:

```
from faker import Faker
import happybase
import random
import hashlib # not used here but could be useful for rowkey hashing (e.g.
reverse domain).
#generating the hash value
def hash_prefix(s: str, length: int = 2) -> str:
    # Returns the first `length` hex digits of the SHA-256 hash of the input
    return hashlib.sha256(s.encode()).hexdigest()[:length]

fake = Faker()
#connects to the hbase server (must be running)
conn = happybase.Connection(host='localhost', port=9090)  # Update host if
needed
conn.open()

table = conn.table('webpages')

domains = ['example.com', 'test.org', 'site.net', 'demo.io', 'sample.co']
html_sizes = ['<p>short</p>', '<div>' + 'medium content ' * 20 + '</div>',
'<section>' + 'large content ' * 100 + '</section>']
```

```
# Track inlinks and outlinks
page_urls = []

# Generate pages
for i in range(20):
#picks a random domain , generates a fake slug and construcs the rowkey as
domain/page-slug
    domain = random.choice(domains)
    slug = fake.slug()
    raw_key = f"{domain}/{slug}"
    prefix = hash_prefix(raw_key)  # e.g., "a7", "d4"
    rowkey = f"{prefix}-{raw_key}"  # final rowkey: "a7-example.com/some-page"

#randomly picks page content size, generates a fake title, Generates a
realistic modified date within the past 120 days, Assigns a fake HTTP status.
    content = random.choice(html_sizes)
    title = fake.sentence()
    last_modified = fake.date_time_between(start_date='-120d',
end_date='now').isoformat()
    status_code = random.choice(['200', '404', '500'])

    # Generate outlinks to random previous pages
    outlinks = random.sample(page_urls, k=min(len(page_urls),
random.randint(0, 3)))
    page_urls.append(rowkey)

    # Add inlink reference for existing pages
    #For every outlink this page has, add a corresponding **inlink** in the
other page
    for outlink in outlinks:
        table.put(outlink, {
            b'inlinks:from': rowkey.encode()
        })

    # Insert the current page into hbase
    table.put(rowkey, {
        b'content:html': content.encode(),
        b'metadata:title': title.encode(),
        b'metadata:status': status_code.encode(),
        b'metadata:last_modified': last_modified.encode(),
        b'metadata:content_size': str(len(content)).encode(),
        **{f'outlinks:to{j}'.encode(): link.encode() for j, link in
enumerate(outlinks)}
    })
```

```
print("✅ Inserted 20 sample web pages.")
```

- run these steps to generate the data:

```
cd shared-data/
hbase shell createtable.txt
hbase thrift start &
python3 generate.py
```

Part2: Business Access Patterns

Business Requirement 1:

Content Management The content team needs to:

- Retrieve the latest version of any page by URL

```
get 'webpages', 'f4-example.com/help-level'
```

```
- this retrieves all the latest columns
```

- View historical versions of a page to track changes

```
get 'webpages', 'f4-example.com/help-level', {COLUMN => 'content:html',
VERSIONS => 3}
```

- List all pages from a specific domain for content audits

```
scan 'webpages', {
  FILTER => "SingleColumnValueFilter('metadata', 'domain', =,
'binary:example.com')"
}
```

- Find all pages modified within a specific time range

```
scan 'webpages', {
  FILTER => "SingleColumnValueFilter('metadata', 'last_modified', >=,
```

```
'binary:2025-05-01')"
}
```

Business Requirement 2:

SEO Analysis The SEO team needs to:

- Find all pages linking to a specific URL (inbound links)

```
scan 'webpages', {
  FILTER => "ValueFilter(=, 'binary:f4-example.com/help-level')"
}
```

```
Took 0.0105 seconds
hbase:110:0> scan 'webpages', {
hbase:111:1*   FILTER => "ValueFilter(=, 'binary:f4-example.com/help-level')"
}
ROW                                COLUMN+CELL
 45-example.com/likely-return-not  column=outlinks:to1, timestamp=2025-05-22T16:06:45.748, value=f4-example.com/help-level
 e
1 row(s)
Took 0.0099 seconds
hbase:113:0>
```

- Identify pages with no outbound links (dead ends)

```
scan 'webpages', {
  FILTER => "SkipFilter(FamilyFilter(=, 'binary:outlinks'))"
}
```

- List pages with the most inbound links (popular pages)
- Retrieve pages with specific content in the title or body

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

```
hbase:104:0> scan 'webpages', {
hbase:105:1*   FILTER => "SingleColumnValueFilter('metadata', 'title', =, 'substring:month')"
}
ROW                                COLUMN+CELL
 d7-site.net/young-somebody-gun    column=content:html, timestamp=2025-05-22T16:06:45.697, value=<p>short</p>
 d7-site.net/young-somebody-gun    column=inlinks:from, timestamp=2025-05-22T16:06:45.711, value=45-example.com/likely-return-note
 d7-site.net/young-somebody-gun    column=metadata:content_size, timestamp=2025-05-22T16:06:45.697, value=12
 d7-site.net/young-somebody-gun    column=metadata:last_modified, timestamp=2025-05-22T16:06:45.697, value=2025-04-15T22:28:31.02353
                                   6
 d7-site.net/young-somebody-gun    column=metadata:status, timestamp=2025-05-22T16:06:45.697, value=200
 d7-site.net/young-somebody-gun    column=metadata:title, timestamp=2025-05-22T16:06:45.697, value=Despite month fill ok century.
 d7-site.net/young-somebody-gun    column=outlinks:to0, timestamp=2025-05-22T16:06:45.697, value=d9-demo.io/turn-me-society
 d7-site.net/young-somebody-gun    column=outlinks:to1, timestamp=2025-05-22T16:06:45.697, value=43-site.net/include-remain
 d7-site.net/young-somebody-gun    column=outlinks:to2, timestamp=2025-05-22T16:06:45.697, value=37-demo.io/from-together-kind
1 row(s)
Took 0.0201 seconds
hbase:107:0>
```

Business Requirement 3:

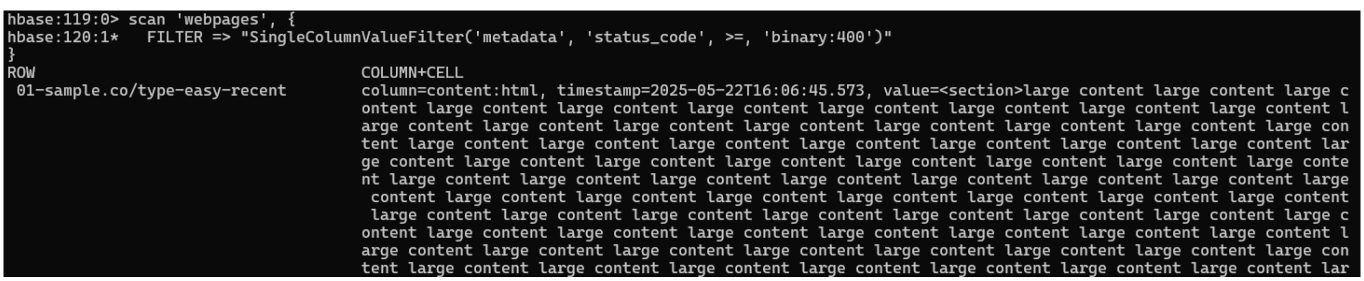
Performance Optimization The performance team needs to:

- Identify the largest pages by content size

```
scan 'webpages', {
  FILTER => "SingleColumnValueFilter('metadata', 'content_size', >,
'binary:10000')"
}
```

- Find pages with HTTP error status codes

```
scan 'webpages', {
  FILTER => "SingleColumnValueFilter('metadata', 'status_code', >=,
'binary:400')"
}
```



- List pages with outdated content (not modified in last 30 days)

```
scan 'webpages', {
  FILTER => "SingleColumnValueFilter('metadata', 'last_modified', <=,
'binary:2025-04-22')"
}
```

Task 3.1: Basic Operations Implement

HBase shell commands to:

- Insert complete web page data (content, metadata, links)

```
put 'webpages', 'a3-example.com/about-us', 'content:html', '<html><body>About
Us</body></html>'
put 'webpages', 'a3-example.com/about-us', 'metadata:title', 'About Us'
put 'webpages', 'a3-example.com/about-us', 'metadata:status', '200'
put 'webpages', 'a3-example.com/about-us', 'metadata:content_size', '1234'
put 'webpages', 'a3-example.com/about-us', 'metadata:last_modified', '2025-05-
20T12:00:00.000Z'
put 'webpages', 'a3-example.com/about-us', 'outlinks:to0', 'b1-
site.net/contact'
put 'webpages', 'a3-example.com/about-us', 'outlinks:to1', 'c2-
demo.io/welcome'
```

```
and to ensure:
get 'webpages', 'a3-example.com/about-us'
```

- Retrieve a page by exact URL

```
get 'webpages', 'a3-example.com/about-us'
```

```
- and if only you know the domain and slug:
```

```
scan 'webpages', {
  FILTER => "RowFilter(=, 'substring:example.com/about-us')"
}
```

- Update a page's content and metadata

```
put 'webpages', 'a3-example.com/about-us', 'content:html', '<html>
<body>Updated content</body></html>'
put 'webpages', 'a3-example.com/about-us', 'metadata:title', 'Updated About Us
Page'
put 'webpages', 'a3-example.com/about-us', 'metadata:status', '200'
put 'webpages', 'a3-example.com/about-us', 'metadata:last_modified', '2025-05-
22T15:30:00.000Z'
put 'webpages', 'a3-example.com/about-us', 'metadata:content_size', '1024'
```

- Delete a page and all its information

```
deleteall 'webpages', 'a3-example.com/about-us'
```

Task 3.2: Filtering Operations

Implement HBase shell commands with filters to:

- Find pages with titles containing specific keywords

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

- Retrieve pages with content size above a threshold

```
scan 'webpages', {
  FILTER => "SingleColumnValueFilter('metadata', 'content_size', >,
'binary:10000')"
}
```

```
 - HBase stores everything as **byte arrays** under the hood.
- `"binary:404"` tells the filter to **do a byte-wise exact match** with the
value `404`.
- It avoids surprises due to how lexicographical (string) comparisons behave
- List pages with specific HTTP status codes
```

```
scan 'webpages', {
  FILTER => "SingleColumnValueFilter('metadata', 'status', =, 'binary:200')"
}
```

- Find pages modified after a specific date

```
scan 'webpages', {
  FILTER => "SingleColumnValueFilter('metadata', 'last_modified', >,
'binary:2025-05-01T00:00:00')"
}
```

Task 3.3: Scanning with Pagination

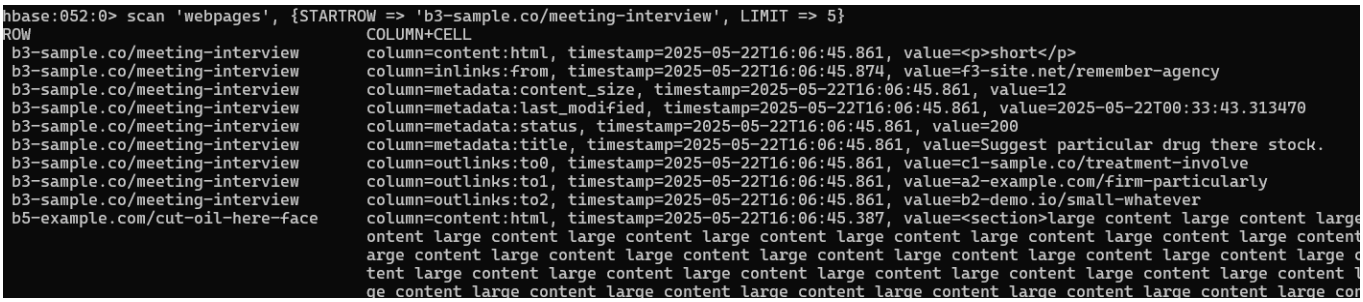
Implement pagination mechanisms to:

- **Pagination** is the process of retrieving a large dataset in smaller, manageable chunks (or pages), rather than loading everything at once.

requirements:

- Scan domain pages in batches of 5 records

```
scan 'webpages', {LIMIT => 5}
```



- Retrieve large result sets efficiently

```
scan 'webpages', {STARTROW => 'b2-demo.io/small-whatever ', FILTER =>
"PageFilter(5)"}
```

```
hbase:054:0> scan 'webpages', {STARTROW => 'b2-demo.io/small-whatever ', FILTER => "PageFilter(5)"}
ROW                                  COLUMN+CELL
 b3-sample.co/meeting-interview      column=content:html, timestamp=2025-05-22T16:06:45.861, value=<p>short</p>
 b3-sample.co/meeting-interview      column=inlinks:from, timestamp=2025-05-22T16:06:45.874, value=f3-site.net/remember-agency
 b3-sample.co/meeting-interview      column=metadata:content_size, timestamp=2025-05-22T16:06:45.861, value=12
 b3-sample.co/meeting-interview      column=metadata:last_modified, timestamp=2025-05-22T16:06:45.861, value=2025-05-22T00:33:43.313470
 b3-sample.co/meeting-interview      column=metadata:status, timestamp=2025-05-22T16:06:45.861, value=200
 b3-sample.co/meeting-interview      column=metadata:title, timestamp=2025-05-22T16:06:45.861, value=Suggest particular drug there stock.
 b3-sample.co/meeting-interview      column=outlinks:to0, timestamp=2025-05-22T16:06:45.861, value=c1-sample.co/treatment-involve
 b3-sample.co/meeting-interview      column=outlinks:to1, timestamp=2025-05-22T16:06:45.861, value=a2-example.com/firm-particularly
 b3-sample.co/meeting-interview      column=outlinks:to2, timestamp=2025-05-22T16:06:45.861, value=b2-demo.io/small-whatever
 b5-example.com/cut-oil-here-face    column=content:html, timestamp=2025-05-22T16:06:45.387, value=<section>large content large content large c
                                     ontent large content large content large content large content large content large content l
                                     arge content large content large content large content large content large content large con
                                     tent large content large content large content large content large content large content lar
                                     ge content large content large content large content large content large content large conte
```

- Implement "next page" functionality using row key markers

```
scan 'webpages', {STARTROW => 'd9-demo.io/turn-me-society\000', LIMIT => 5}
```

```
    -this rowkey represents the last row key in previous page, and we use the
null byte to trick hbase to move just past the current row
```

```
hbase:055:0> scan 'webpages', {STARTROW => 'd9-demo.io/turn-me-society\000', LIMIT => 5}
ROW                                  COLUMN+CELL
 e1-test.org/parent-apply-common     column=content:html, timestamp=2025-05-22T16:06:45.357, value=<p>short</p>
 e1-test.org/parent-apply-common     column=inlinks:from, timestamp=2025-05-22T16:06:45.930, value=82-demo.io/result-blood-policy
 e1-test.org/parent-apply-common     column=metadata:content_size, timestamp=2025-05-22T16:06:45.357, value=12
 e1-test.org/parent-apply-common     column=metadata:last_modified, timestamp=2025-05-22T16:06:45.357, value=2025-04-03T17:07:22.0
07485
 e1-test.org/parent-apply-common     column=metadata:status, timestamp=2025-05-22T16:06:45.357, value=404
 e1-test.org/parent-apply-common     column=metadata:title, timestamp=2025-05-22T16:06:45.357, value=Against talk although eat.
 e1-test.org/parent-apply-common     column=outlinks:to0, timestamp=2025-05-22T16:06:45.357, value=59-example.com/detail-start-har
d
 e1-test.org/parent-apply-common     column=outlinks:to1, timestamp=2025-05-22T16:06:45.357, value=37-demo.io/from-together-kind
 e1-test.org/parent-apply-common     column=outlinks:to2, timestamp=2025-05-22T16:06:45.357, value=1a-site.net/traditional-trial
 e5-test.org/hundred-democrat        column=content:html, timestamp=2025-05-22T16:06:45.792, value=<div>medium content medium cont
ent medium co
                                     ntent medium content medium content medium content medium content medium content medium conte
nt medium con
```

- 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

```
get 'webpages', 'd9-demo.io/turn-me-society', {COLUMN => 'content:html',
VERSIONS => 3}
```

```
Took 0.0057 seconds
hbase:061:0> get 'webpages', 'f4-example.com/help-level', {COLUMN => 'content:html', VERSIONS => 3}
COLUMN                               CELL
 content:html                        timestamp=2025-05-22T17:26:41.735, value=<html><body>Version 3 - Final Help</body></html>
 content:html                        timestamp=2025-05-22T17:26:33.847, value=<html><body>Version 2 - Updated Help</body></html>
 content:html                        timestamp=2025-05-22T17:26:28.770, value=<html><body>Version 1 - Help</body></html>
1 row(s)
Took 0.0125 seconds
hbase:062:0>
```

- Demonstrate how TTL automatically removes old content

    - When you read data, HBase checks the age of each version. If it's older than TTL, it's ignored. during periodic compaction, HBase physically removes expired data from storage
- Implement a manual purge for outdated content

```
delete 'webpages', 'e1-test.org/parent-apply-common', 'content:html',
TIMESTAMP => 1715600000000
```

```
hbase:065:0> delete 'webpages', 'e1-test.org/parent-apply-common', 'content:html', TIMESTAMP => 1715600000000
Took 0.0055 seconds
hbase:066:0>
```

- Show how to retrieve the latest N versions of content

```
get 'webpages', 'f4-example.com/help-level', {COLUMN => 'content:html',
VERSIONS => 2}
```

```
hbase:066:0> get 'webpages', 'f4-example.com/help-level', {COLUMN => 'content:html', VERSIONS => 2}
COLUMN                          CELL
 content:html                   timestamp=2025-05-22T17:26:33.847, value=<html><body>Version 2 - Updated Help</body></html>
 content:html                   timestamp=2025-05-22T17:26:28.770, value=<html><body>Version 1 - Help</body></html>
1 row(s)
Took 0.0244 seconds
hbase:067:0>
```