Reporting Tables Optimization

Sprint 2: NewspaperFeeds, MagazineFeeds, Sentiment_Analysis_keywords, SentimentAnlyResult

1. Overview

This report explains the structure and relationships between the tables NewspaperFeeds, MagazineFeeds, Sentiment_Analysis_keywords, SentimentAnlyResult. It describes what each table stores, the indexes they use, what issues were found, and how to improve them.

2. NewspaperFeeds

The **NewspaperFeeds** table keeps information about each newspaper article, such as the link of the article, name of the newspaper and link of the newspaper. It is a master list of all the newspaper articles.

Column Name	Data Type	Description
URL	VARCHAR(1000) (Primary Key)	A unique link to the article. (Not NULL)
NewspaperName	VARCHAR(150)	The name of the newspaper.
NewspaperLink	VARCHAR(150)	Link of the newspaper in which the article was published

Indexes

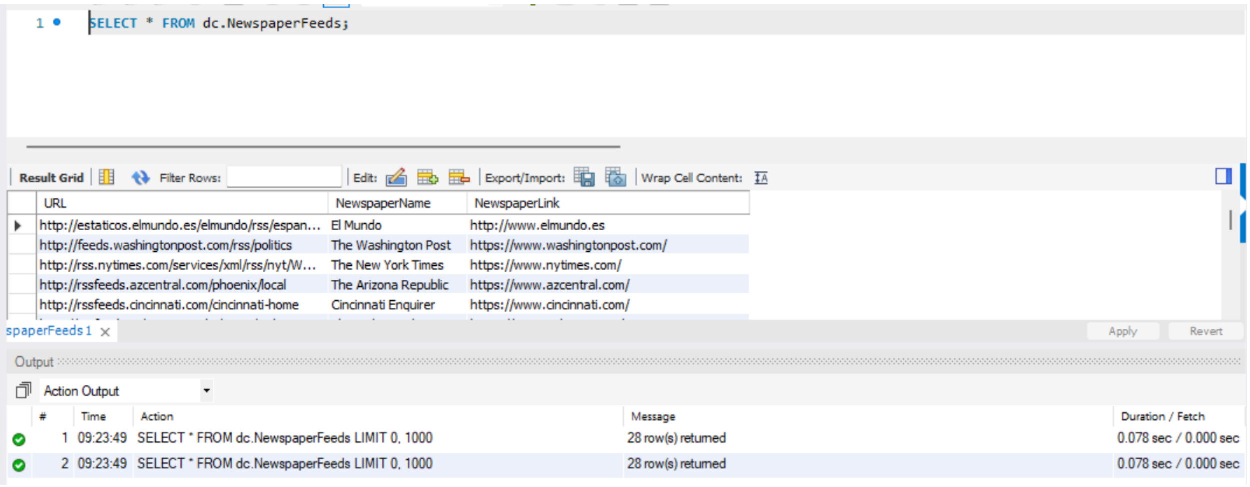
Index Name	Type	Purpose
PRIMARY	PRIMARY, BTree on URL	The main index automatically created on the URL column. The link is unique and is searchable.

URL_UNIQUE	BTree on URL	Adds a uniqueness to the url column

Observations / Issues

1. **The primary key being the url**
   URLs are long text fields and using them as primary keys leads to slow indexing and higher memory usage.
2. **URL_UNIQUE index on the same column as the PRIMARY**
   The PRIMARY KEY already implies uniqueness. The UNIQUE constraint adds a duplicate index enforcing the same rule.
3. **No foreign keys. The table is not connected to anything.**
   It is very hard to get information about this table and no other tables can access this table neither for receiving or storing data. The data is isolated.

Data query:



```sql
SELECT * FROM dc.NewspaperFeeds;
```

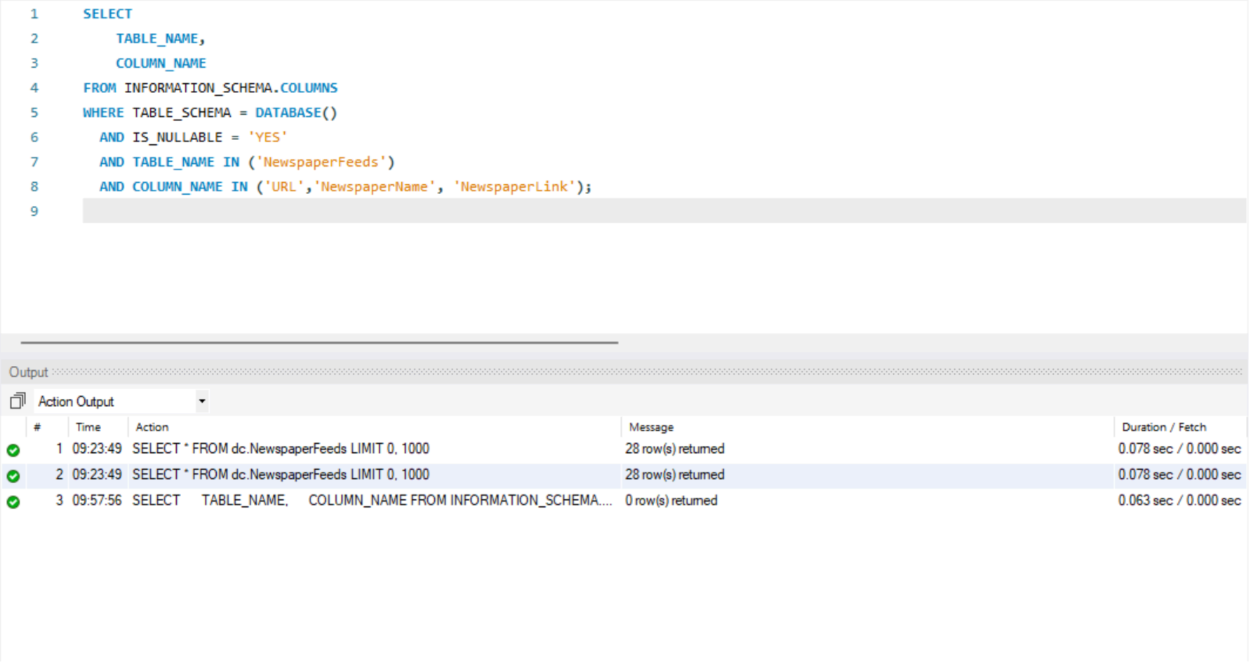
URL	NewspaperName	NewspaperLink
http://estaticos.elmundo.es/elmundo/rss/espan...	El Mundo	http://www.elmundo.es
http://feeds.washingtonpost.com/rss/politics	The Washington Post	https://www.washingtonpost.com/
http://rss.nytimes.com/services/xml/rss/nyt/W...	The New York Times	https://www.nytimes.com/
http://rssfeeds.azcentral.com/phoenix/local	The Arizona Republic	https://www.azcentral.com/
http://rssfeeds.cincinnati.com/cincinnati-home	Cincinnati Enquirer	https://www.cincinnati.com/

#	Time	Action	Message	Duration / Fetch
1	09:23:49	SELECT * FROM dc.NewspaperFeeds LIMIT 0, 1000	28 row(s) returned	0.078 sec / 0.000 sec
2	09:23:49	SELECT * FROM dc.NewspaperFeeds LIMIT 0, 1000	28 row(s) returned	0.078 sec / 0.000 sec

Observation: The query worked and returned 28 rows in 0.078

Issue: There are only 28 rows of data in this table and because the data is isolated most of its links are outdated.

Null and Missing values query:



```sql
SELECT
    TABLE_NAME,
    COLUMN_NAME
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = DATABASE()
  AND IS_NULLABLE = 'YES'
  AND TABLE_NAME IN ('NewspaperFeeds')
  AND COLUMN_NAME IN ('URL','NewspaperName', 'NewspaperLink');
```

```
   #  Time      Action                                                        Message              Duration / Fetch
   1  09:23:49  SELECT * FROM dc.NewspaperFeeds LIMIT 0, 1000                 28 row(s) returned   0.078 sec / 0.000 sec
   2  09:23:49  SELECT * FROM dc.NewspaperFeeds LIMIT 0, 1000                 28 row(s) returned   0.078 sec / 0.000 sec
   3  09:57:56  SELECT     TABLE_NAME,     COLUMN_NAME FROM INFORMATION_SCHEMA....   0 row(s) returned    0.063 sec / 0.000 sec
```

Recommendations

- URL_UNIQUE index can be dropped
- Reconsider the usage of URL as the PRIMARY index
- Reoptimize the connection between the tables and add foreign keys accordingly.

3. MagazineFeeds

Purpose

The **MagazineFeeds** table keeps information about each magazine article, such as the link of the article, name of the magazine and link of the magazine. It is a master list of all the magazine articles.

Column Name	Data Type	Description
URL	VARCHAR(1000) (Primary Key)	A unique link to the article. (Not NULL)

MagazineName	VARCHAR(150)	The name of the magazine.
MagazineLink	VARCHAR(150)	Link of the magazine in which the article was published

Indexes

Index Name	Type	Purpose
PRIMARY	PRIMARY, BTree on URL	The main index automatically created on the URL column. The link is unique and is searchable.
URL_UNIQUE	BTree on URL	Adds a uniqueness to the url column

Observations / Issues

1. **The primary key being the url**
   URLs are long text fields and using them as primary keys leads to slow indexing and higher memory usage.
2. **URL_UNIQUE index on the same column as the PRIMARY**
   The PRIMARY KEY already implies uniqueness. The UNIQUE constraint adds a duplicate index enforcing the same rule.
3. **No foreign keys. The table is not connected to anything.**
   It is very hard to get information about this table and no other tables can access this table neither for receiving or storing data. The data is isolated.

Recommendations

- URL_UNIQUE index can be dropped
- Reconsider the usage of URL as the PRIMARY index
- Reoptimize the connection between the tables and add foreign keys accordingly.

4. Sentiment_Analysis_Keywords

Purpose

The **Sentiment_Analysis_Keywords** table keeps information about the content of the media, especially the keywords associated with them. It has ID, keyword, and submission time

Column Name	Data Type	Description
ID	INT	A unique id assigned to these keywords
Keyword	VARCHAR(20)	The keywords associated with it.
Submission_Time	datetime	Time when the content was analyzed

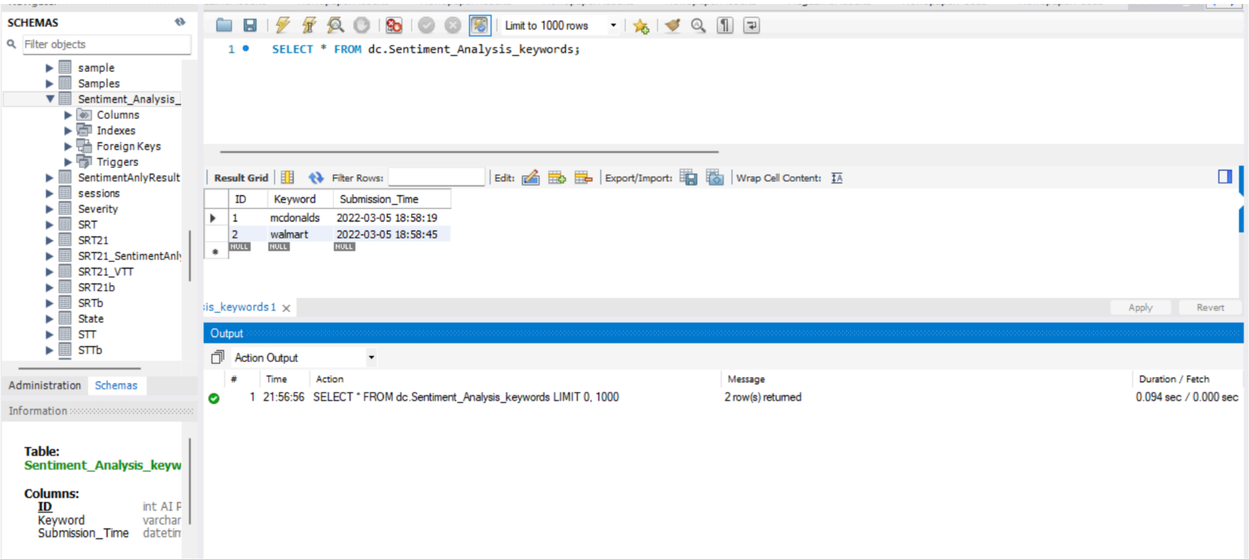
Indexes

Index Name	Type	Purpose
ID	PRIMARY	The main index automatically created on the ID column.

Observations / Issues

1. **No relationships with source tables:**
   The table is not linked to any feed table (e.g., NewspaperFeeds or MagazineFeeds). As a result, it's unclear which article or content each keyword belongs to.
2. **Limited keyword size:**
   The Keyword column uses VARCHAR(20), which may be too short to store multi-word or compound keywords used in sentiment analysis.
4. **No indexing on Keyword or Submission_Time:**
   Queries filtering by keyword or analyzing time-based patterns will perform slowly without indexes on these columns.
5. **No foreign key constraints:**
   Without foreign keys to the analyzed content (article or feed), there's no referential integrity between keywords and their sources.

Data query:



```
SELECT * FROM dc.Sentiment_Analysis_keywords;
```

ID	Keyword	Submission_Time
1	mcdonalds	2022-03-05 18:58:19
2	walmart	2022-03-05 18:58:45
NULL	NULL	NULL

```
#  Time      Action                                                   Message            Duration / Fetch
1  21:56:56  SELECT * FROM dc.Sentiment_Analysis_keywords LIMIT 0, 1000  2 row(s) returned  0.094 sec / 0.000 sec
```

Table: Sentiment_Analysis_keyw

Columns:

ID	int AI P
Keyword	varchar
Submission_Time	datetin

No other tests can be run on this data.

Recommendations

- Add a FeedID or ArticleID column that references either NewspaperFeeds or MagazineFeeds depending on the source.
- Add indexes on Keyword and Submission_Time to improve query performance.
- Increase the Keyword column size to VARCHAR(100) for flexibility.
- Add NOT NULL constraints to Keyword and Submission_Time.

4. SentimentAnlyResult

Purpose

The **SentimentAnlyResult** table keeps information about the results of the sentiment analysis done on media content (specifically radio).

Column Name	Data Type	Description
Station	Text	Name or identifier of the radio station from which the analyzed content originated. Used to group or filter sentiment results by source.

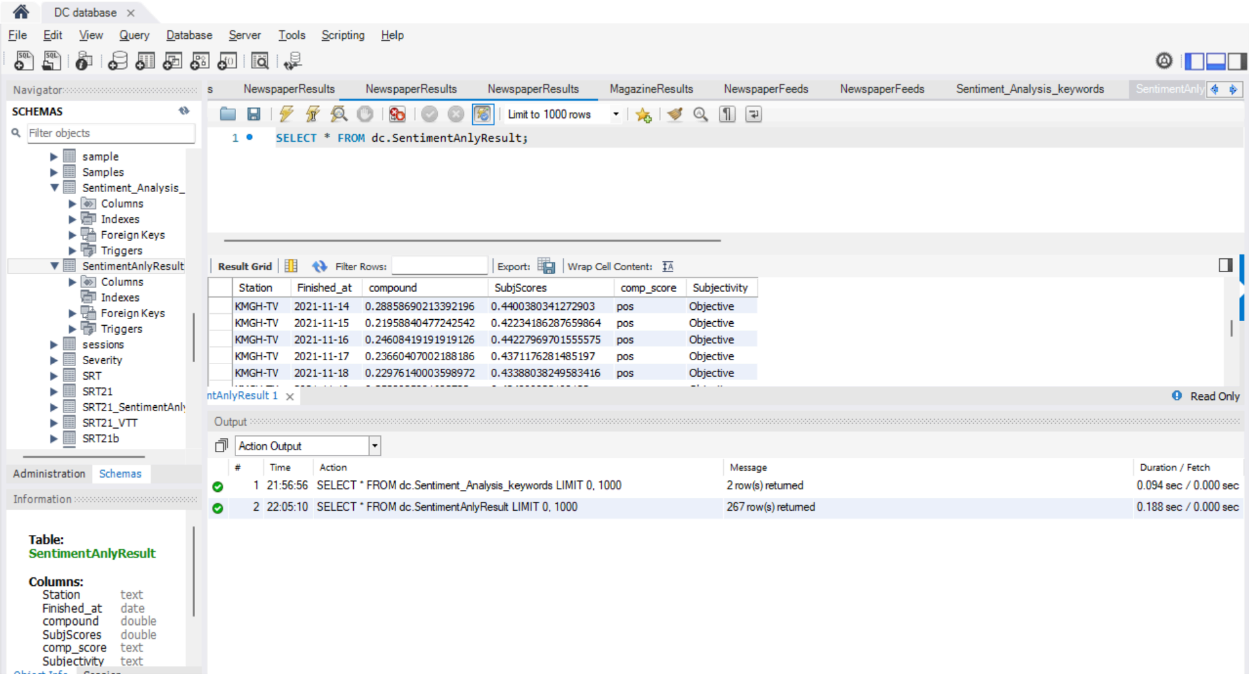
Finished_at	date	The date when the sentiment analysis process for this record was completed. Useful for tracking analysis timelines or historical trends.
compound	double	The **overall sentiment score** calculated by the sentiment analysis algorithm. Typically ranges from -1 (most negative) to +1 (most positive).
SubjScores	double	The **subjectivity score**, representing how subjective or opinionated the content is (e.g., 0 = objective, 1 = highly subjective).
comp_scores	text	Raw or component sentiment scores (e.g., individual positive/negative/neutral values) stored as text or JSON for detailed inspection.
Subjectivity	text	Qualitative label representing the level of subjectivity (Objective, Subjective, Mixed). Often derived from the SubjScores value.

Indexes: Has no indexes

Observations / Issues

1. **No relationship with feed tables:**
   The Station field is stored as text and does not reference NewspaperFeeds or MagazineFeeds. This breaks data integrity and makes it impossible to determine which feed each sentiment result belongs to.
2. **Lack of indexing:**
   The table has no indexes, which will degrade performance as the number of records grows.
3. **Data type inconsistencies:**
   Fields like comp_scores and Subjectivity are stored as text but contain structured or numerical data that could be stored as JSON or numeric types.

Data query:



```
SELECT * FROM dc.SentimentAnlyResult;
```

Station	Finished_at	compound	SubjScores	comp_score	Subjectivity
KMGH-TV	2021-11-14	0.28858690213392196	0.4400380341272903	pos	Objective
KMGH-TV	2021-11-15	0.21958840477242542	0.42234186287659864	pos	Objective
KMGH-TV	2021-11-16	0.24608419191919126	0.44227969701555575	pos	Objective
KMGH-TV	2021-11-17	0.23660407002188186	0.4371176281485197	pos	Objective
KMGH-TV	2021-11-18	0.22976140003598972	0.43388038249583416	pos	Objective

#	Time	Action	Message	Duration / Fetch
1	21:56:56	SELECT * FROM dc.Sentiment_Analysis_keywords LIMIT 0, 1000	2 row(s) returned	0.094 sec / 0.000 sec
2	22:05:10	SELECT * FROM dc.SentimentAnlyResult LIMIT 0, 1000	267 row(s) returned	0.188 sec / 0.000 sec

Table: SentimentAnlyResult

Columns:

Station	text
Finished_at	date
compound	double
SubjScores	double
comp_score	text
Subjectivity	text

No other tests can be run on this data.

Recommendations

- Add a FeedID column referencing NewspaperFeeds or MagazineFeeds to associate results with articles.
- Add indexes on FeedID and Finished_at to optimize filtering and lookups.
- Convert comp_scores to JSON and Subjectivity to ENUM ('Objective', 'Subjective', 'Mixed').
- Add NOT NULL constraints to key columns (FeedID, compound, Finished_at).
- Add foreign key relationships to Sentiment_Analysis_Keywords to link results with the keywords used.
- Add created_at and updated_at timestamps.

Move **Categories** into its own table for better organization and easier searches.

5. Relationship Mapping

Current Structure

Currently, none of the reporting tables (NewspaperFeeds, MagazineFeeds, Sentiment_Analysis_Keywords, SentimentAnlyResult) are linked. Each table functions in isolation, storing useful but disconnected data. This prevents cross-analysis (e.g., viewing sentiment trends for specific feeds or correlating keywords with sources).

Recommended Structure

- Create a unified relationship structure where feeds (newspaper and magazine) act as the core data sources, and sentiment/keyword tables depend on them.
- NewspaperFeeds and MagazineFeeds: Act as master content sources.
- Sentiment_Analysis_Keywords: Links to feed tables through FeedID (foreign key).
- SentimentAnlyResult: Links to both FeedID (source article) and KeywordID (keyword used).

One-to-Many Relationships:

- One article (in NewspaperFeeds or MagazineFeeds) → Many sentiment keywords
- One keyword → Many sentiment results.
- One article → Many sentiment results (indirectly via keywords).

Benefits of This Structure

- Enables sentiment analysis across all article types.
- Ensures referential integrity (no orphan sentiment or keyword entries).
- Simplifies analytical queries (e.g., sentiment trends by publisher or keyword).
- Improves maintainability and scalability as data volume grows.

6. Data Type and Index Improvements

- Use INT AUTO_INCREMENT for all ID fields.
- Convert URL fields from primary to unique indexes.
- Change Finished_at and Submission_Time to DATETIME.
- Replace long TEXT fields with VARCHAR(255) where appropriate.
- Use ENUM or JSON for categorical and structured sentiment data.

Indexes to Add

- On FeedID in Sentiment_Analysis_Keywords and SentimentAnlyResult
- On Keyword in Sentiment_Analysis_Keywords
- On Finished_at in SentimentAnlyResult
- On ArticleID in both feed tables

7. Recommendations Summary

1. Introduce surrogate keys (ArticleID, FeedID, KeywordID, ResultID).
2. Establish foreign key relationships among all four tables.
3. Convert text timestamps to DATETIME.
4. Add NOT NULL constraints on essential columns.
5. Simplify indexes by keeping only the most relevant.
6. Normalize repeated large text fields.

11. Conclusion

The reporting tables provide useful data independently but lack the relational structure needed for efficient cross analysis. By establishing foreign key relationships, and optimizing data types and indexes, the database can support faster queries, better integrity, and scalable analytics across newspaper and magazine content.