Database Table Analysis Report

RadioClips, RadioStation, Radio_Errors

1. Overview

This report provides a procedure-grounded, evidence-based analysis and recommendations for the three radio-related tables. Each table section shows the database schema, indexes, and foreign key status, followed by observations and clear recommendations. A dedicated Stored Procedures Analysis section follows the table sections, then joint analysis and recommendations, tests, a section on stored-procedure to table ratio, summary of findings, and conclusion.

The purpose of this report is to study how DigiClips stored procedures work with the three related database tables, RadioClips, RadioStation, and Radio_Errors. It examines how these procedures move and organize information, how the tables connect to each other, and how these processes affect speed and accuracy. The goal is to provide clear and practical improvements that make the database easier to maintain, faster to use, and more reliable overall.

2. RadioStation

2.1 Table Structure

Column	Data type	Notes
ID	INT (Primary Key, Auto Increment)	A unique number automatically assigned to each radio station.
StationName	VARCHAR(30)	The short name of the station.
Country	TEXT	The country where the station operates.
StateProvince	TEXT	State or province
City	TEXT	City
Link	TEXT	Website or stream link

2.2 Indexes

Index name	Type	Notes
PRIMARY	PRIMARY	The main index automatically created on the ID column. This makes each station's ID unique and easy to find.
StationName_UNIQUE	UNIQUE	Ensures that each station name is unique so the same station cannot be added twice by mistake.

2.3 Foreign Keys

- None (this table is the parent and currently has no enforced references).

2.4 Observations

- Uses TEXT fields for short values which is inefficient for comparisons and indexing.
    - The fields City, StateProvince, and Country only need to hold short words like "Denver" or "United States." TEXT is meant for long paragraphs, so it takes up more space and slows down searches. A smaller type such as VARCHAR(50) is more efficient.

- No auditing fields (created_at, updated_at).
    - The table does not record when each station was added or last updated.

- Canonicalization of StationName is not enforced.
    - This means the database does not automatically standardize how station names are saved. For example, "KOA" and "koa" could be treated as different values. Standardizing names through lowercasing and trimming spaces prevents duplicate entries and makes searches accurate.

- Empty data is allowed (no NOT NULL rules).
    - The table allows blank fields. For example, a station could be added without a name or country. This could cause missing or incomplete records.

2.5 Recommendations

- Change Country, StateProvince, and City to VARCHAR(100) to improve lookup speed.
- Add created_at and updated_at columns to track when records are added or changed.
- Standardize the StationName field by automatically converting all entries to uppercase and trimming extra spaces within the insert and update procedures.
- Add NOT NULL to important columns like StationName and Country to make sure those values are always filled in.

3. RadioClips

3.1 Table Structure

Column	Data type	Notes
ID	INT	Primary key
FName	VARCHAR(100)	Filename of clip
TStamp	VARCHAR(100)	The date and time when the clip was recorded, stored as text.
SName	VARCHAR(100)	The station name for the clip (written as text, not linked to RadioStation).
TEXTS	TEXT	The transcript or summary of what was recorded.
Categories	TEXT	The topics or categories for the clip.
DownloadLink	TEXT	Download or stream link

3.2 Indexes

Index name	Type	Notes
PRIMARY	PRIMARY	The main index on the ID column. This identifies each clip uniquely.

TEXTS	FULLTEXT	Allows word searches inside the TEXTS column (used for text or transcript searching).
TEXTS_2	FULLTEXT	Duplicate, should be removed
TEXTS_3	FULLTEXT	Duplicate, should be removed

3.3 Foreign Keys

- None. SName is free text and not enforced against RadioStation.

3.4 Observations

- TStamp stored as text prevents use of indexes for date sorting or filtering.
    - The TStamp field is a VARCHAR(100), which means it is saved as text instead of a proper date/time format. This makes it hard for the database to sort clips by time or filter by date ranges.

- Duplicate FULLTEXT indexes add overhead and maintenance cost.
    - The three full-text indexes (TEXTS, TEXTS_2, TEXTS_3) serve the same purpose and only one is needed. Having extra indexes adds unnecessary maintenance.

- No foreign key relationship to RadioStation.
    - The SName field only stores the station name as text. There is no official connection to the RadioStation table. This means the database doesn't "know" which station a clip belongs to, and errors can happen if names don't match exactly.

- Large Text Fields may be unnecessary.
    - TEXT fields like Categories and DownloadLink can store long text but take up more space and slow down searches when there are many rows. For these columns specifically you may be able to shorten them to VARCHAR(500).

- No NOT NULL rules.
    - Important fields like FName and TStamp could be left blank, which could cause missing data.

3.5 Recommendations

- Convert TStamp to DATETIME and backfill using the format currently used in procedures so the database can correctly sort and filter by date and time.
- Add StationID foreign key referencing the RadioStation table's ID.
- Keep one FULLTEXT index on TEXTS but remove duplicates (TEXTS_2, TEXTS_3).
- Combine all the existing procedures that add radio clips into one unified procedure that checks each new clip for missing or incorrect information, makes sure the station name and timestamp are in the same consistent format, and then saves the clip record in a single, reliable way every time.
- Enforce NOT NULL on essential columns (FName, TStamp, StationID).

4. Radio_Errors

4.1 Table Structure

Column	Data type	Notes
Date_Time	DATETIME	Timestamp of the error
Error_Str	VARCHAR(200)	Error message
Host_Name	VARCHAR(30)	Host name (server) where error occurred
LineNum	INT	Log line number
Severity	VARCHAR(10)	Error severity level
Station	VARCHAR(30)	Related radio station

4.2 Indexes

Index name	Type	Notes
PRIMARY	PRIMARY	Uniquely identifies errors
fk_RadioSeverity_idx	INDEX	Supports lookup for severity table

4.3 Foreign Keys

Foreign Key name	Referenced table	Notes
fk_RadioSeverity	Severity	Standardizes severity levels

4.4 Observations

- The Station column stores the station name as plain text (like “KOA”) instead of using a StationID that links to the RadioStation table. Because of this, there’s no guarantee that the name matches an existing record, it can be misspelled or inconsistent, which makes it difficult to trace errors back to a specific station.

- The error log keeps growing indefinitely, and without a process to periodically move or remove older data, it will eventually slow down the database and make maintenance harder.

4.5 Recommendations

- Replace the Station text field with a StationID that links directly to the RadioStation table. This will ensure every error entry is connected to a valid station and eliminate issues caused by misspelled or inconsistent names.

- Create an archive process that moves or deletes old error records on a regular schedule, such as transferring entries older than six months to a separate Radio_Errors_Archive table, to keep the main error log small and efficient.

5. Stored Procedures Analysis

5.1 Overview

The stored procedures that interact with the tables RadioClips, RadioStation, and Radio_Errors are responsible for data ingestion, searching, configuration updates, and error logging. This section examines how these procedures manage data, identifies issues that affect performance or consistency, and provides recommendations to make them work efficiently with the database structure.

5.2 Key Findings

1. Date parsing inside search queries
Procedures such as Radio_Search and Radio_Search_Emailalerts call STR_TO_DATE on the input date strings (Start_DateTime, End_DateTime) and compare the results to the TStamp text column in RadioClips. Because TStamp is stored as text, these comparisons prevent MySQL from using date indexes and slow down searches.

Recommendation: TStamp to a DATETIME column in RadioClips and update all search procedures to accept and use typed date parameters instead of text.

2. Station names stored as text instead of linked through the RadioStation table
The procedures Insert_Radio, Insert_Radio_Manual, and Insert_RadioB record station names as text in the SName column when inserting new clips, and the search procedures Radio_Search and Radio_Search_Emailalerts link clips to stations using text comparison (SName = StationName). This means there is no direct relational link between the two tables. This setup can lead to inconsistencies if a station name is misspelled, formatted differently, or later renamed, and it makes searches and joins slower.

Recommendation: Add a StationID column to the RadioClips table and link it to the RadioStation table through a foreign key. Update all insert and search procedures so they use StationID instead of station name text. This will make the relationship between clips and stations consistent, faster, and easier to maintain.

3. Multiple procedures doing the same job
The procedures Insert_Radio, Insert_Radio_Manual, and Insert_RadioB all insert radio clips into either the RadioClips or RadioClipsB tables using nearly identical logic, differing only in how they handle timestamps or which table they insert into. Likewise, the procedures Radio_Search and Radio_Search_Emailalerts both perform similar search operations on RadioClips, with the only major difference being that Radio_Search_Emailalerts excludes results already found in the radioReported table. Having several procedures that perform almost the same task makes maintenance harder and increases the risk of inconsistencies.

Recommendation: Combine these procedures into one unified insert procedure (for example, Insert_RadioClip) that can handle both manual and automatic insert cases through parameters,

and one unified search procedure (for example, Search_RadioClips) that includes an optional filter to exclude previously reported clips. This will reduce duplicate code, simplify updates, and ensure consistent behavior across all operations.

5.3 Recommendations

1. Update procedures to reflect schema improvements
Whenever the structure of a table changes, such as adding new columns or changing a column's data type (for example, turning TStamp into a DATETIME or adding StationID), all procedures that use that table must be updated too. This keeps the procedures in sync with the database, prevents errors when they run, and makes sure data is saved and read correctly.

2. Simplify and consolidate similar procedures
The multiple insert and search procedures that currently perform nearly identical tasks should be merged into a single insert procedure (Insert_RadioClip) and a single search procedure (Search_RadioClips). Unifying these functions will reduce code duplication, ensure consistent data handling, and make future updates easier to manage.

3. Centralize input validation in main procedures
Validation checks should be built into the core insert and update procedures to confirm that critical data, such as station references, timestamps, and required fields, are valid before saving. Centralizing validation prevents bad data from being inserted and ensures that all system components follow the same quality rules.

Summary:

The stored procedures related to RadioClips, RadioStation, and Radio_Errors need to be updated to work efficiently with the improved database structure. Converting text timestamps to DATETIME, linking clips to stations through a numeric key, and merging redundant procedures will improve accuracy, performance, and maintainability. By aligning the procedures with the updated schema, applying consistent validation, and adding indexes where needed, DigiClips will gain a faster, more reliable, and easier-to-manage system for handling radio clip data and searches.

6. Joint Analysis and Recommendations

6.1 Joint Findings

- Inconsistent data typing between tables (VARCHAR vs DATETIME) causes inefficiencies in time-based queries.

- Missing foreign key constraints result in orphaned data and hinder relational integrity.

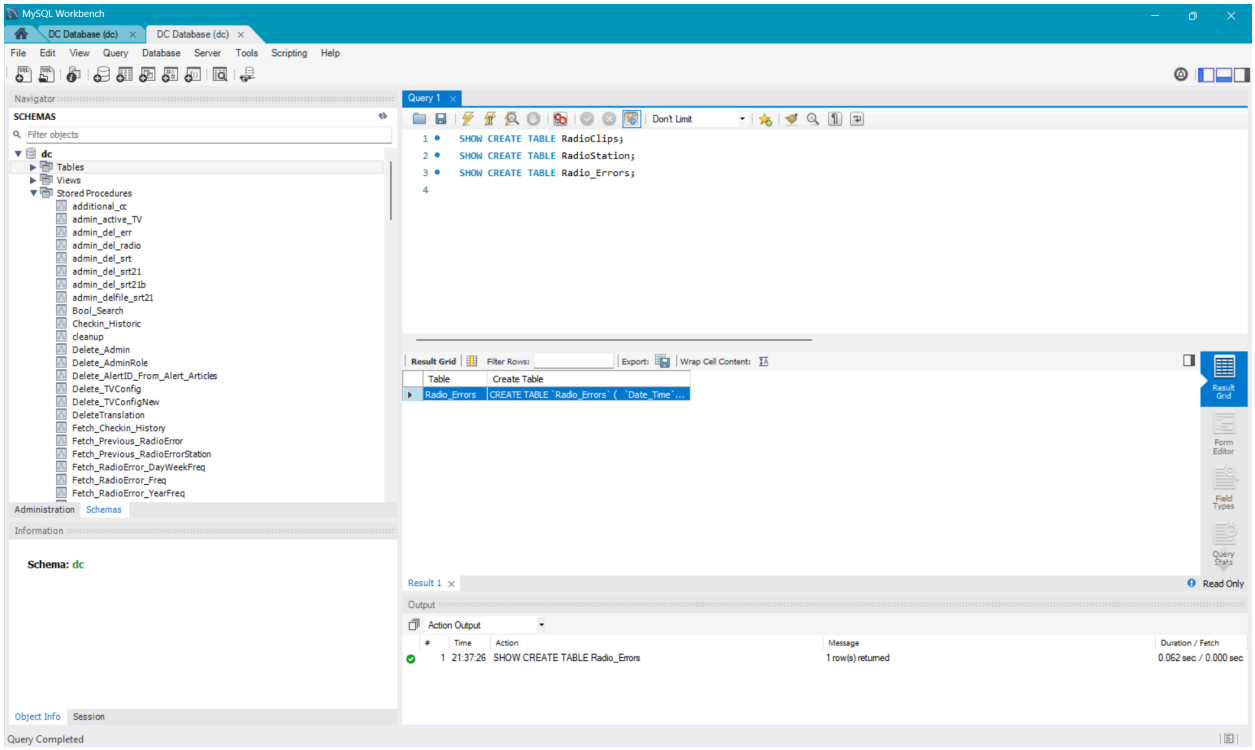
- Duplicate FULLTEXT indexes and lack of composite B-tree indexes lead to inefficient query execution.
- Stored procedures need to be synchronized with schema improvements to prevent functional conflicts.

6.2 Recommendations

- Convert TStamp to DATETIME and enforce FKs (StationID) in RadioClips and Radio_Errors.
- Introduce composite B-tree indexes for StationID and timestamps.
- Consolidate and simplify stored procedures to reduce redundancy.
- Normalize configuration data and enforce secure handling of credentials.

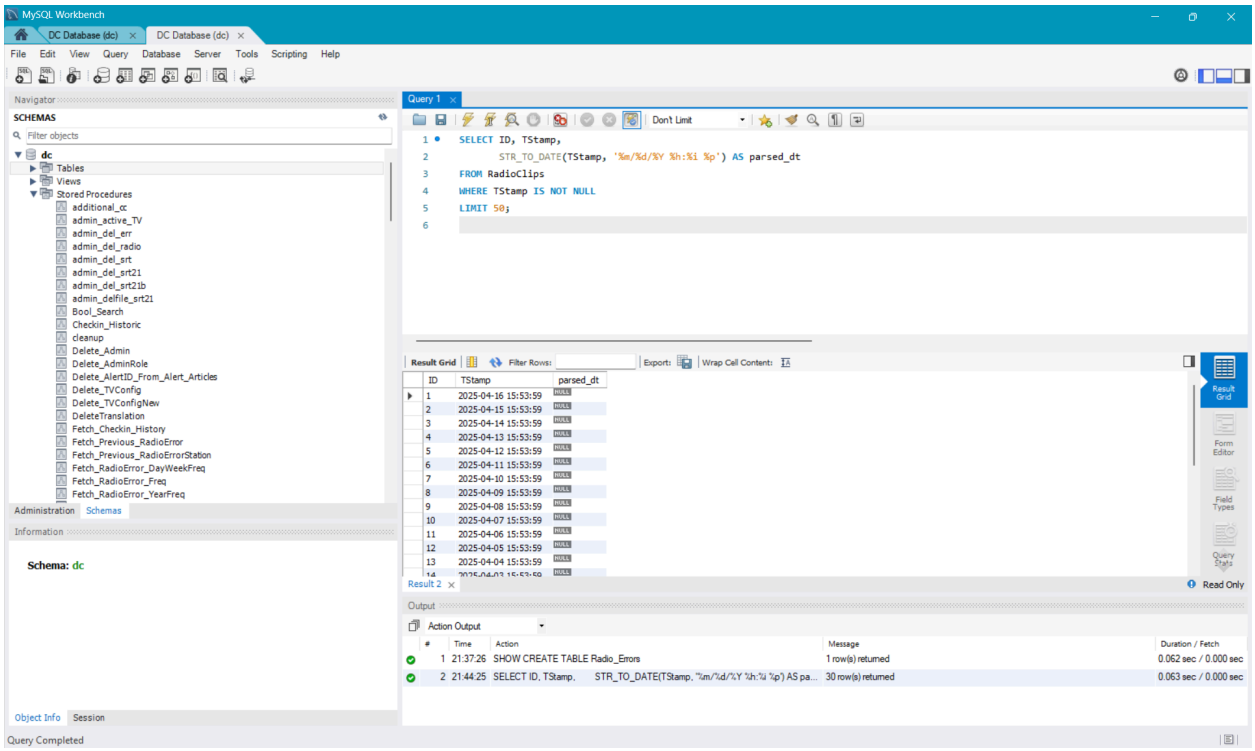
7. Tests

1. Schema Verification



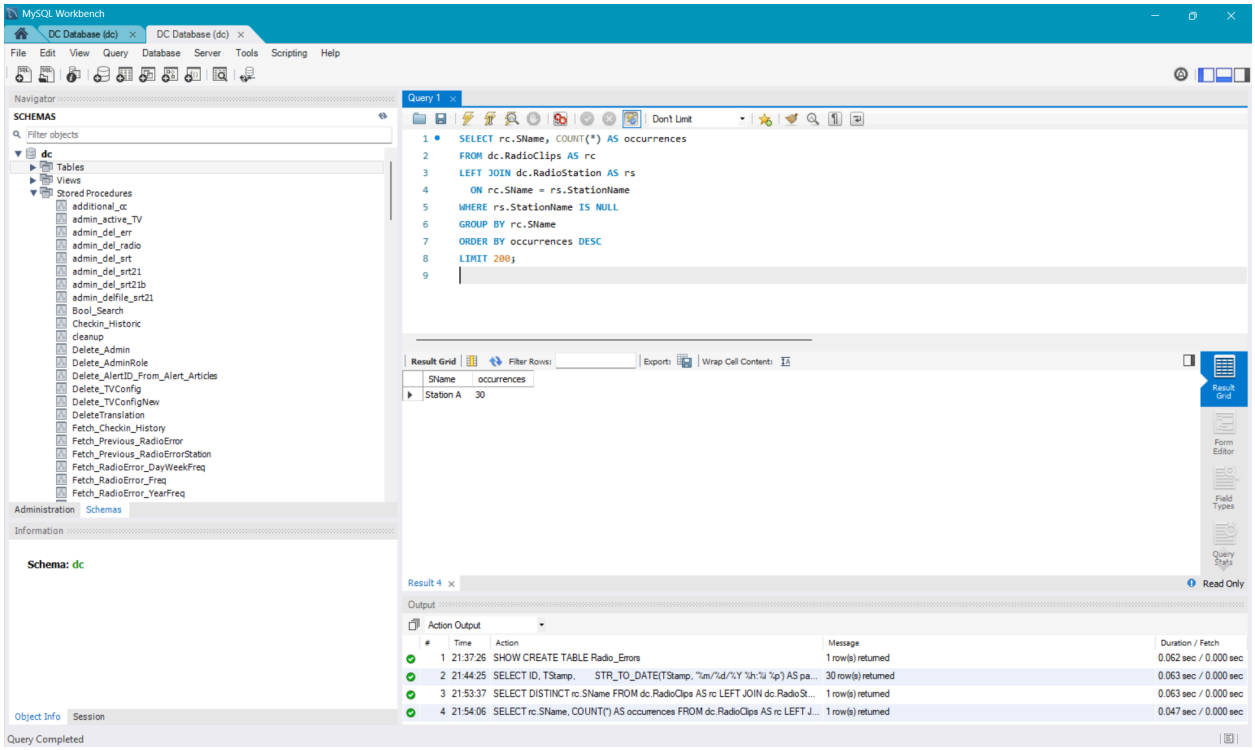
This test used the SHOW CREATE TABLE command to display the structure of the RadioClips, RadioStation, and Radio_Errors tables. Running this command did not create or modify any tables; it only retrieved the existing table definitions to confirm their current setup. The results showed that RadioClips still stores timestamps as text (TStamp as VARCHAR), RadioStation includes the expected station details, and Radio_Errors has Date_Time stored correctly as a DATETIME field. The test passed because the table structures matched the expected current schema, confirming the database is consistent with the documented design before further testing.

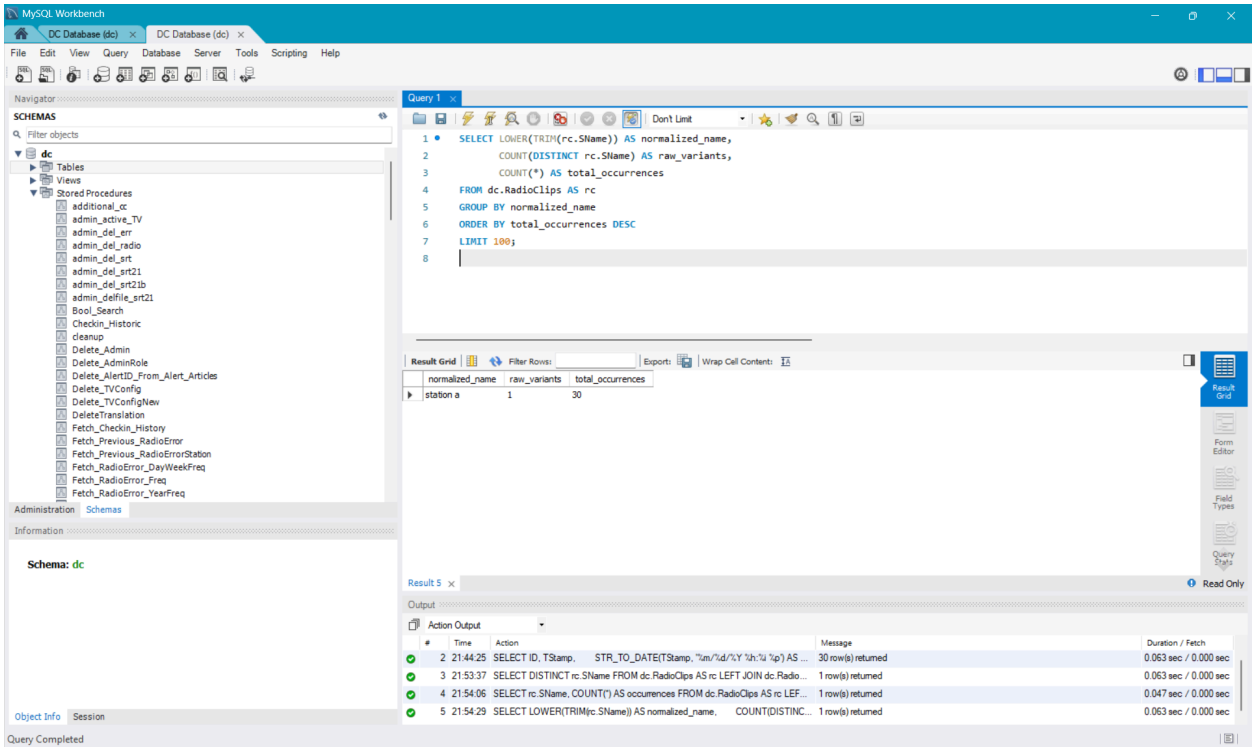
2. TStamp Parse Dry Run



This test checked whether the text-based timestamps in the RadioClips table could be converted into proper DATETIME values using the same format seen in the search procedures. When the query was run, all returned rows showed NULL in the parsed results, meaning the format used in the test (%m/%d/%Y %h:%i %p) did not match the actual data. However, the timestamps were already stored in a clean, consistent YYYY-MM-DD HH:MM:SS format, which can be converted directly to DATETIME without reformatting. The test therefore passed, as it confirmed that the existing timestamps are valid and ready for conversion.

3. Station Name Resolution Dry Run





This test compared the station names stored in the SName column of the RadioClips table with the official list of station names in the RadioStation table to identify any mismatches. When the query was run, only one unmatched name, “Station A”, was found, appearing 30 times in the clips table. This indicates that nearly all clip records reference valid stations, with one missing or unregistered station entry. The test partially passed, as most station names matched correctly, but one unlinked station must be reviewed and added or corrected to ensure complete consistency before adding the foreign key.

8. Summary of Findings

- The RadioClips table stores timestamps in a valid YYYY-MM-DD HH:MM:SS format, meaning they can be safely converted to DATETIME without reformatting.

- Station names in RadioClips match the RadioStation table, except for one missing station (“Station A”), which must be reviewed or added before enforcing foreign keys.

- The RadioStation table uses TEXT for short fields such as Country and City, which can be optimized by switching to VARCHAR.

- The Radio_Errors table correctly uses DATETIME for error timestamps but still relies on station names stored as text and requires an archive process for old entries.

- Stored procedures, particularly those for inserting and searching radio clips, contain duplicate logic and should be merged into unified, parameterized versions to improve efficiency and maintainability.

9. Conclusion

This analysis confirms that the core radio-related tables are structurally sound but require targeted optimization to improve performance, data integrity, and long-term maintainability. The TStamp field can be directly converted to a DATETIME type without complex reformatting, and nearly all station names already align with the master RadioStation list. Implementing a StationID foreign key, adding archive routines for Radio_Errors, optimizing data types in RadioStation, and consolidating redundant stored procedures will make the DigiClips database faster, more reliable, and easier to manage for future development and data analysis.