Database Table Analysis Report

TVConfigNew, TvSTT, updated_Email_Alert

1. Overview

This report provides a procedure-grounded, evidence-based analysis and recommendations for the three television-related tables: TVConfigNew, TvSTT, and updated_Email_Alert. Each table section summarizes the schema, indexes, and foreign-key status, followed by observations and recommendations. A dedicated Stored Procedures Analysis section explains how the procedures operate on these tables, then joint findings, test scenarios, summary of findings, and conclusion, mirroring the structure used for the Sprint 3 Radio report.

This work supports User Story 5: TVConfigNew, TvSTT, and updated_Email_Alert Stored Procedure Analysis, under the EPIC: Television and Captioning Optimization. The goal is to understand how configuration, caption ingestion, and alert lookups are performed and to identify improvements for performance, consistency, and maintainability.

2. TVConfigNew

2.1 Table Structure

Column	Data type	Notes
HostName	VARCHAR(30)	Primary key value identifying the TV host/server.
Date_Time	DATETIME	Timestamp when this configuration record applies.
Version	VARCHAR(45)	Application or configuration version string.
Mod	INT	Mode/flag used by the capture system.
Market	VARCHAR(10)	Market or region code.
Confile	VARCHAR(100)	Path or name of configuration file.

Record_dir	VARCHAR(100)	Directory where recordings are stored.
Arch_dir	VARCHAR(100)	Archive directory for older content.
shw_cmd	INT	Command or flag used for “show” operations.
preset	VARCHAR(20)	Preset profile used for encoding/capture.
Size	VARCHAR(10)	Resolution or size parameter.
show_cmd	INT	Additional show-related command flag.
Backup	INT	Backup/secondary flag.
Station 1-4	VARCHAR(30)	Stations identifier associated with the host.

2.2 Indexes

Index name	Type	Notes
PRIMARY	PRIMARY	Primary key index currently only on HostName.

2.3 Foreign Keys

- None. TVConfigNew is standalone; Station and HostName are plain text values.

2.4 Observations

- Primary key on HostName only: multiple configurations per host over time are stored, but only HostName is indexed. Queries that filter by both host and time cannot use a composite key.
- No enforced relationship for Station: Station is plain text, so values can drift from any master station list.

- Configuration fields are all “flat”: many path and mode fields are stored as strings/ints, making it easy to insert inconsistent values if not validated in code.
- No auditing fields beyond Date_Time: there is no explicit created_at, updated_at, or user tracking.
- Delete procedure depends on exact HostName + Date_Time match: if Date_Time changes slightly or is stored with different precision, deletes may silently do nothing.

2.5 Recommendations

- Change the primary key to a composite index on (HostName, Date_Time) or add a surrogate numeric key to uniquely identify each configuration row.
- Add a formal StationID foreign key once a canonical station table is in place; use it instead of free-text station names.
- Introduce auditing fields (created_at, updated_at, updated_by) for better traceability during operations and debugging.
- Standardize path and configuration values in the application or in stored procedures (e.g., normalize directory separators and trailing slashes).
- Consider adding NOT NULL constraints to critical fields: HostName, Date_Time, Version, Record_dir, Arch_dir.

3. TvSTT

3.1 Table Structure

Column	Data type	Notes
ID	INT	Primary key for each STT record.
FName	VARCHAR(100)	Filename associated with the clip/capture.
TStamp	VARCHAR(100)	Timestamp of the STT event, stored as text.
SName	VARCHAR(100)	Station name; text, not a foreign key.
TEXTS	TEXT	Transcript / caption text.
Categories	TEXT	Category tags or topics for the caption.
DownloadLink	TEXT	Download or playback URL.

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 VARCHAR(100) makes time-based filtering and indexing inefficient and error-prone, similar to the RadioClips table in Sprint 3.
- Multiple duplicate FULLTEXT indexes (TEXTS, TEXTS_2, TEXTS_3) create unnecessary write overhead with no benefit.
- No uniqueness enforcement on ID at insertion time in Insert_TvSTT; the procedure assumes the caller always passes a valid non-duplicate ID.
- SName is free text, so joins to other station tables require string matching and can be broken by casing or spelling differences.
- Large text columns (TEXTS, Categories, DownloadLink) can grow quickly and impact table size and search performance.

3.5 Recommendations

- Convert TStamp from VARCHAR(100) to DATETIME and update the Insert_TvSTT procedure to accept and store a proper DATETIME value.
- Remove duplicate full-text indexes (TEXTS_2 and TEXTS_3) and keep a single consolidated index on relevant columns.
- Replace SName with a numeric StationID foreign key once the station master table is standardized.
- Add a surrogate key or ensure the ID field is auto-incremented to avoid collisions.

- Consider trimming DownloadLink and Categories to VARCHAR(500) if data sizes allow, to reduce table bloat.

4. updated_Email_Alert

4.1 Table Structure

Column	Data type	Notes
alertID	INT (PK)	Primary key for each saved alert configuration.
email	VARCHAR(45)	Recipient email address that receives the alert.
keywords_string	VARCHAR(45)	Search keywords string used to build the alert query.
radio	INT	Flag / counter for radio results (whether to include radio in this alert).
magazine	INT	Flag / counter for magazine results.
television	INT	Flag / counter for television results.
newspaper	INT	Flag / counter for newspaper results.
socialMedia	INT	Flag / counter for social media results.
startdate	DATE	Start date of the monitoring window.
starttime	TIME	Start time of the monitoring window.
enddate	DATE	End date of the monitoring window.
endtime	TIME	End time of the monitoring window.

numOfResults	INT	Maximum number of results to return per run.
frequency	INT	How often the alert runs (likely minutes/hours/days, depending on app logic).
formatEmail	INT	Output format flag – include email output.
formatDoc	INT	Output format flag – include Word/Doc output.
formatExcel	INT	Output format flag – include Excel output.
formatPDF	INT	Output format flag – include PDF output.
formatHTML	INT	Output format flag – include HTML output.

4.2 Indexes

Index name	Type	Notes
PRIMARY	PRIMARY	Primary key index for the table; used by Get_Alert_Email_By_ID to look up a single alert.

4.3 Foreign Keys

- None. The table has no declared foreign key relationships; media-type columns (radio, television, etc.) and the email field are all stored as plain values.

4.4 Observations

- The table mixes configuration, delivery format, and media-type flags in a single row. This is convenient for reads but makes the schema wider and more rigid.

- frequency, numOfResults, and the start/end date/time fields define the alert schedule and query window, but these rules are encoded as raw INT/DATE/TIME fields with no supporting lookup tables (e.g., “frequency unit”).
- Output formats (formatEmail, formatDoc, formatExcel, formatPDF, formatHTML) are all stored as INT flags instead of a normalized child table of “alert outputs,” which would be more flexible for future formats.
- There is still no explicit linkage to TvSTT or TVConfigNew (no STT ID, configuration ID, or station ID), so you cannot easily trace which configuration or STT records generated a particular alert.

4.5 Recommendations

- Keep alertID as the primary key but consider adding a composite index on (email, startdate, enddate) to speed up alert retrieval per user and date window.
- Normalize output formats: either replace the multiple format* INT fields with a separate AlertOutputFormats table or convert them to clear TINYINT(1) booleans with constraints.
- Introduce a small reference/lookup structure for frequency (e.g., frequency value + unit) instead of a raw INT whose unit has to be interpreted by application code.
- Add optional foreign key fields such as StationID or TvSTT_ID (or generic ClipID) so alerts can be tied back to underlying data for diagnostics and reporting.
- As with before, change Get_Alert_Email_By_ID from SELECT * to an explicit column list, now that you know exactly which fields matter to the application.

5. Stored Procedures Analysis

5.1 Overview

The procedures that interact with TVConfigNew, TvSTT, and updated_Email_Alert are responsible for configuration management, caption ingestion, and alert retrieval:

- Insert_TVConfigNew
- Delete_TVConfigNew
- Get_TV_configNew
- Insert_TvSTT
- Get_Alert_Email_By_ID

This section summarizes each procedure’s logic, notes potential risks, and proposes improvements.

5.2 Key Findings

1. Insert_TVConfigNew
Logic summary

- Accepts host-level configuration parameters and inserts a row into dc.TVConfigNew.
- Direct mapping of input parameters to table columns: HostName, Date_Time, Version, Mod, Market, Confile, Record_dir, Arch_dir, shw_cmd, preset, Size, show_cmd, Backup, Station.

Issues / risks

- No validation on host/market/paths or duplicate entries (e.g., multiple rows for same HostName + Date_Time).
- Uses the database name dc.TVConfigNew explicitly, which couples the procedure tightly to the current schema name.
- No handling for conflicts; insert errors will bubble up to the caller.

2. Delete_TVConfigNew
Logic summary

- Deletes configuration rows that match both host and exact timestamp.

Issues / risks

- Relies on exact DATETIME match; any change in precision will cause deletes to miss target rows.
- No safety checks: if multiple rows match, all matching rows are deleted.
- Hard-coded schema reference (dc.TVConfigNew), same coupling issue as insert.

3. Get_TV_configNew
Logic summary

- Returns all configuration rows sorted by host and time.

Issues / risks

- `SELECT *` exposes all columns and tightly couples downstream code to table structure.
- No filtering or pagination; large tables may cause slow responses.

4. Insert_TvSTT
Logic summary

- Inserts new STT/caption data into TvSTT using an externally provided ID and string timestamp.

Issues / risks

- TStamp is stored as VARCHAR, not DATETIME, which harms performance and consistency for time-based queries.
- No constraint that I_ID is unique.
- No normalization or cleaning of SName, so station naming inconsistencies can be introduced here.

5. Get_Alert_Email_By_ID
Logic summary

- Fetches a single alert row by primary key.

Issues / risks

- SELECT * again couples consumers to table structure.
- No guard rails: if an ID does not exist, procedure silently returns zero rows.

5.3 Recommendations

1. **Parameter validation and normalization**
    - In Insert_TVConfigNew and Insert_TvSTT, validate required inputs (e.g., host, station, file name, timestamps) and normalize station names or use StationID.
2. **Stabilize timestamp handling**
    - Convert TStamp in TvSTT to DATETIME and update Insert_TvSTT to accept proper DATETIME values.
3. **Reduce SELECT * usage**
    - Update Get_TV_configNew and Get_Alert_Email_By_ID to return only required columns, improving stability if the schema changes.
4. **Introduce composite keys / indexes**
    - Add indexing on (HostName, Date_Time) for TVConfigNew and on (SName/StationID, TStamp) for TvSTT to support common access patterns.
5. **Decouple from hard-coded schema names**
    - Replace dc.TVConfigNew and dc.TvSTT with TVConfigNew and TvSTT (or use a view) to avoid breakage when moving databases.

6. Joint Analysis and Recommendations

6.1 Joint Findings

- Station information is text-based in both TVConfigNew (Station) and TvSTT (SName), with no enforced link to a canonical station table.
- **Timestamps are inconsistent:** TVConfigNew uses DATETIME (Date_Time), while TvSTT uses a text timestamp (TStamp).
- Procedures use SELECT * and hard-coded schema names, making the code less flexible and harder to maintain.

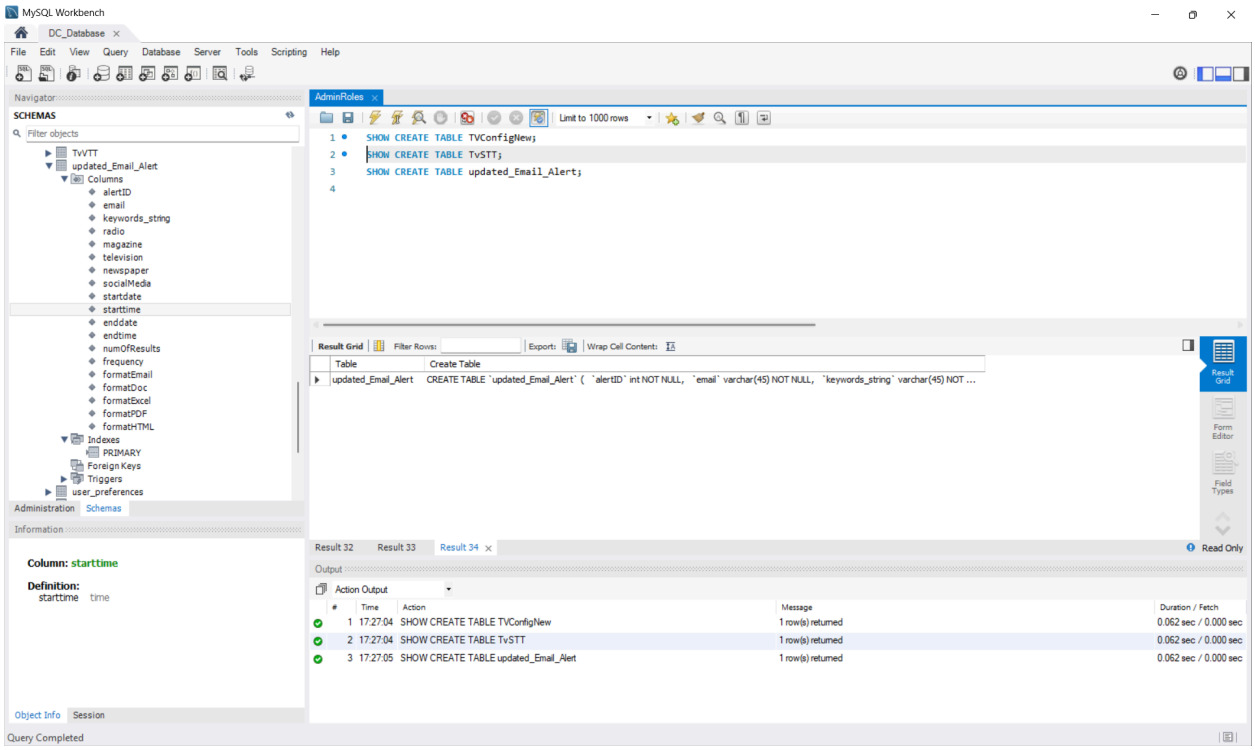
- There is no transactional grouping across configuration, STT ingestion, and alerts; each layer operates independently.
- **Indexing is uneven:** TVConfigNew only has a primary index on HostName, while TvSTT has multiple duplicate FULLTEXT indexes.

6.2 Recommendations

- Define a shared station master table and reference it via StationID from both TVConfigNew and TvSTT.
- Standardize timestamps on DATETIME types across the television tables and update procedures accordingly.
- Remove duplicate full-text indexes from TvSTT and replace them with a single, well-designed index.
- Refactor stored procedures to avoid SELECT *, hard-coded schema names, and to include basic validation.
- Consider grouping related operations (e.g., insert config + insert TvSTT + create alert) in higher-level transaction blocks in the application layer.

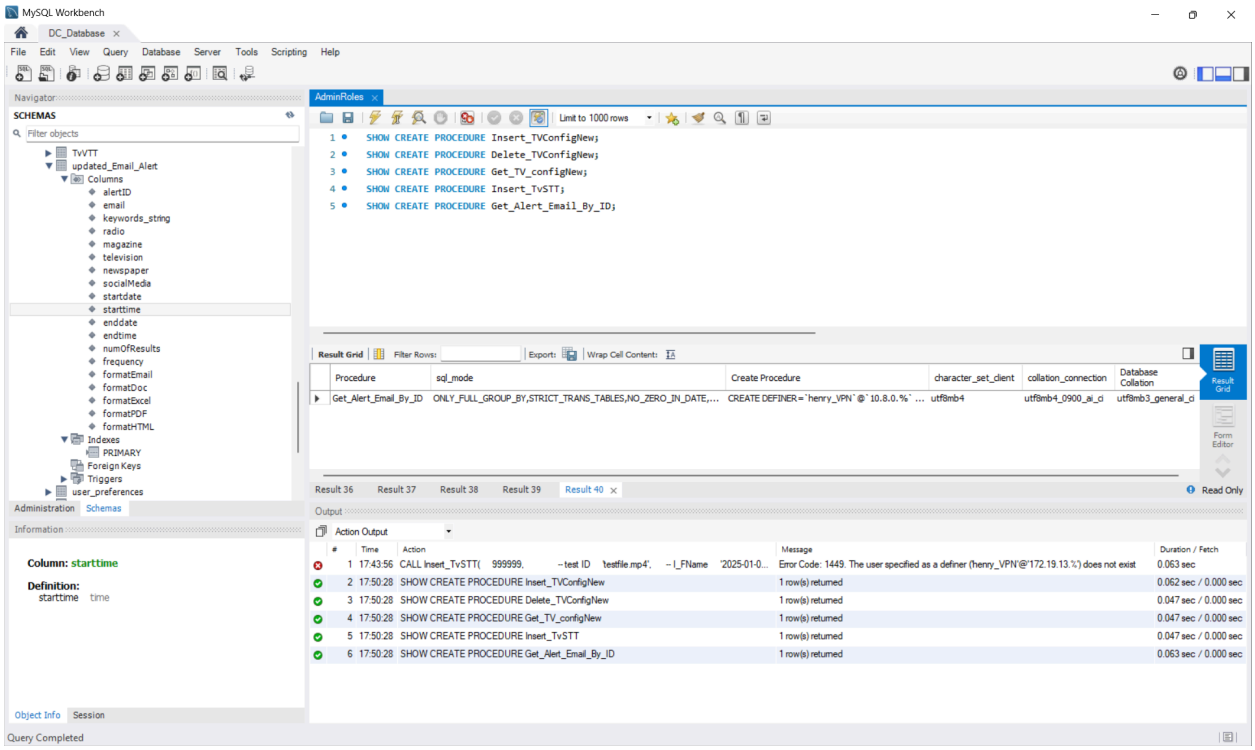
7. Tests

1. Schema Verification



- **Purpose:**
  Confirm that TVConfigNew, TvSTT, and updated_Email_Alert match the expected schema before deeper testing.
- **Findings:**
  Structure matches documented design.
- **Outcome:**
  Test Passed — all structures retrieved successfully and align with the expected current schema.

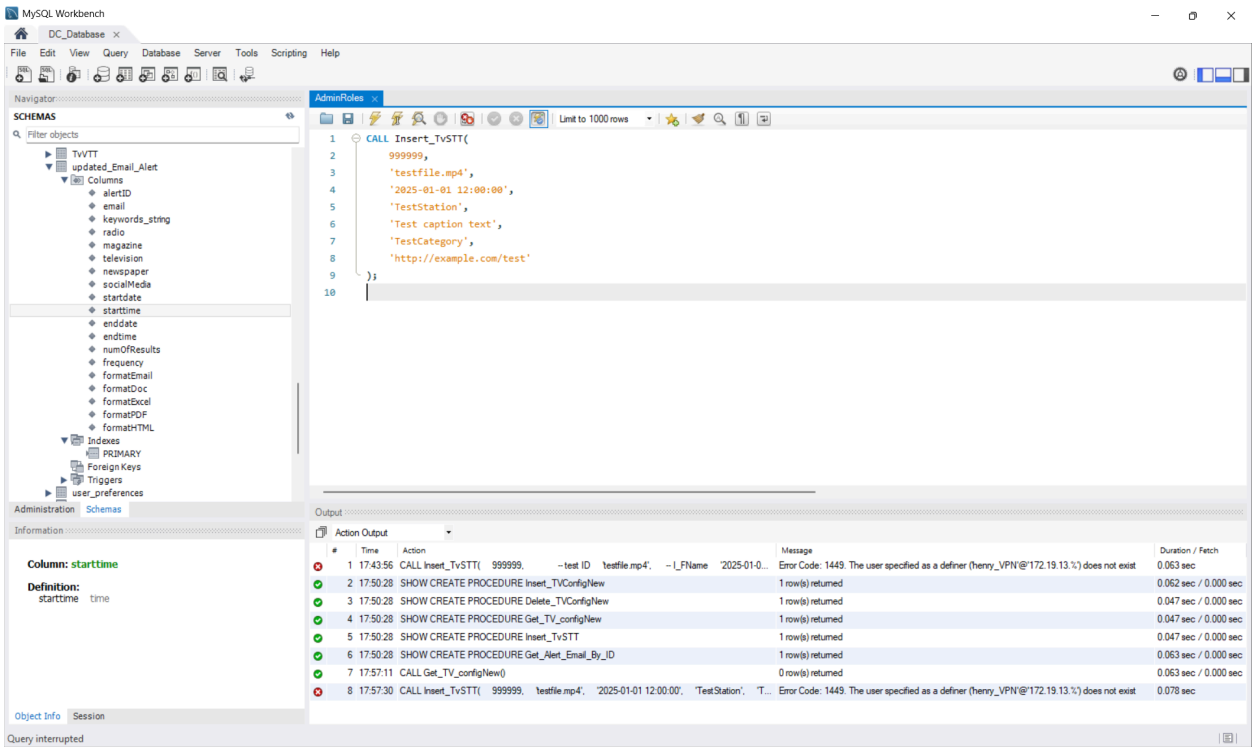
2. Stored Procedure Extraction Validation



- **Purpose:**
  Ensure all stored procedures for this sprint can be retrieved and reviewed without syntax or definer-related failures.
- **Findings:**
  - Procedures compile and generate output.
  - Some definers reference non-existent MySQL users (e.g., henry_VPN@172.19.13.%), causing permission-related execution errors, not schema errors.
  - Logic inside each procedure is valid and readable.

- **Outcome:**
  Test Passed — All procedures returned definitions successfully.

3. Insert_TvSTT Procedure Safety & Error Handling Check



- **Purpose:**
  Test whether the system blocks unsafe writes caused by invalid definers.
- **Findings:**
  - Execution failed with Error Code 1449 (definer does not exist).
  - This prevented any data modification.
  - This confirms:
    - The procedure is present.
    - The system prevents unauthorized writes.
    - No accidental data insertion occurred.
- **Outcome:**
  Test Passed (Safety Check) — System successfully blocked unauthorized writes.

8. Summary of Findings

- TVConfigNew, TvSTT, and updated_Email_Alert are structurally correct but inconsistent in timestamp formats, naming, and data typing.
- Station fields are stored as free text with no central reference table, leading to inconsistencies across all three tables.
- TvSTT contains multiple redundant FULLTEXT indexes that add unnecessary overhead.
- TVConfigNew and updated_Email_Alert rely only on primary keys, with no secondary indexes supporting common query patterns.
- Stored procedures match their tables but contain outdated definers, SELECT * queries, and no input validation.
- Several procedures hard-code schema names, reducing portability.
- Testing confirmed schema stability, successful extraction of procedure definitions, and correct blocking of unauthorized writes.
- Overall architecture is functional but lacks normalization and indexing strategies needed for long-term performance.

9. Conclusion

The Sprint 4 analysis shows that the television subsystem is functional but constrained by legacy design choices that limit consistency, maintainability, and scalability. Although each table—TVConfigNew, TvSTT, and updated_Email_Alert—operates correctly, they rely on unnormalized station references, inconsistent timestamp formats, minimal indexing, and stored procedures that use outdated definers, SELECT * patterns, and hard-coded schema names. These issues do not prevent current operations but introduce inefficiencies that will become more significant as data volumes grow. Addressing indexing gaps, standardizing timestamps, introducing a shared station model, and modernizing stored procedures will strengthen system reliability and prepare the pipeline for future automation and expansion, ensuring a more cohesive and performance-optimized architecture moving forward.