Streaming Service Database System Project Report

Ronaldo Rodriguez, Peter Siri, Andy Yip, Mazen Zwin

San Francisco State University

CSC 675 Intro to Database Systems

Prof. Jose Ortiz

May 12, 2025

Streaming Service Database System: Final Testing & Analysis Report

Project Overview

The project is centered around a MySQL database that models a streaming platform. Here's a summary of the main components:

- *Users:* Registered users with profiles, watchlists, and subscriptions.
- Content: Movies and TV shows, each with associated genres, directors, actors, and countries of origin.
- Watchlist: Users can add content to their watchlists, with a trigger to enforce a maximum of 50 items.
- Ratings and Reviews: Users can rate and review content, which feeds into recommendations and content availability.
- *Transactions:* Tracks user payments and logs any negative transaction attempts.
- Administrative Tables: Includes error logs and tables for tracking problematic assignments (like duplicate directors).

 Scheduled Events: Periodic cleanup and maintenance tasks, such as archiving content with persistently low ratings.

Database Schema

The schema includes tables for users, content, actors, directors, genres, countries, watchlists, reviews, ratings, transactions, and mapping tables for many-to-many relationships. We put a lot of effort into getting the foreign key relationships right, especially since they're critical for the data integrity.

Some highlights:

- Mapping Tables: For things like Content_Genre, Content_Actor, and Content_Director, to support many-to-many relationships.
- *Error Log Table:* Central place for all errors and trigger-logged events.
- *Director_Assignment_Errors:* Tracks attempts to assign the same director to the same content more than once.

ETL Process

The ETL script (etl_load_data.py) is responsible for loading initial data from a CSV file into the database. It uses pandas for data manipulation and SQLAlchemy for database operations.

We made sure to:

- Load data in the right order to respect foreign key constraints.
- Clean up tables before each run to avoid duplicate entries.
- Handle missing or malformed data gracefully.

Triggers and Business Logic

Watchlist Limit Trigger (trg_watchlist_limit)

Ensures users can't have more than 50 items in their watchlist. If a new item is added beyond the limit, the oldest is removed automatically.

• Unique Director Trigger (trg_unique_director)

Prevents assigning the same director to the same content more than once, and logs the attempt.

• Negative Transaction Logging (trg_log_transaction_error)

Catches and logs any attempt to insert a negative transaction amount.

• Archive Content on Low Rating (trg_archive_content_on_low_rating)

If the average rating for a piece of content drops below 2.0, the content is automatically archived and the event is logged.

Common issue: we ran into a lot of syntax errors with triggers at first—especially forgetting to drop old triggers before creating new ones, and sometimes having stray words or lines in the SQL file. Cleaning up the SQL scripts and always using DROP TRIGGER IF EXISTS before each trigger fixed these problems.

Scheduled Events

I set up scheduled events to automate maintenance tasks. For example, there's an event that checks for content with persistently low ratings and archives it, ensuring the platform doesn't promote unpopular content. I verified these by checking the Error_Log table after the events ran.

Results of Testing

Test Case	Input/Action	Expected	Actual Output	Pass/Fai	Issue &
		Output		l	Resolution
Schema	Run schema.sql	All tables	Success after	Pass	Had to add FK
Creation		created, no	disabling FK		disable/enable
		FK errors	checks		for clean drops

ETL Load	Run	Data	Loaded	Pass	Needed clean,
	etl_load_dat	loaded into	successfully		empty tables
	a.py	all tables	after schema cleanup		before ETL
Watchlist	Insert >50 items	Only 50	Works as	Pass	Syntax errors
Limit	for a user	items,	expected		fixed by cleaning
Trigger		oldest			trigger file
		removed			
Unique	Assign same	Error and	Error raised,	Pass	Added DROP
Director	director twice	log entry	log written		TRIGGER IF
Trigger					EXISTS before
					CREATE

Archive on	Insert low ratings	Content	Works as	Pass	Syntax errors
Low Rating	for content	archived,	expected		fixed, ensured
		log entry			triggers dropped
		made			first
Negative	Insert negative	Error	Log entry	Pass	Confirmed
Transaction	transaction	logged in	created		Error_Log entry,
		Error_Lo			fixed syntax
		g			
Calcadulad	Wait for /tuing a	I a a autoi as	I a sa austui an	Duna	VaniG ad lan
Scheduled	Wait for/trigger	Log entries	Log entries	Pass	Verified by
Events	scheduled event	in	confirmed		checking
		Error_Lo			Error_Log table
		g			

Performance Insights

• Query Performance:

Most queries ran quickly, even with joins on mapping tables. I added indexes on key columns (like user_id, content_id) to keep things fast.

• ETL Speed:

The ETL script handled the dataset efficiently. For larger datasets, I'd consider batching inserts or further optimizing indexes.

• Scheduled Events:

No noticeable slowdowns from scheduled events. I monitored the Error_Log table to confirm events executed as expected.

Common Errors and How I Fixed Them

Here are the most persistent issues I encountered, along with actual error messages and solutions:

1. Foreign Key Constraint Errors

Error Message:

ERROR 1217 (23000): Cannot delete or update a parent row: a foreign key constraint fails

Fix:

Added this to schema setup:

SET FOREIGN KEY CHECKS = 0;

-- Drop and create tables

 $SET\ FOREIGN_KEY_CHECKS = 1;$

2. Trigger Already Exists / Trigger Syntax Errors

Error Messages:

Error Code: 1359. Trigger already exists

Error Code: 1064. You have an error in your SQL syntax

0 497 10:06:49 INSERT INTO Content_Genre (content_id, genre_id) VALUES (2, 2)

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails

(`multimediacontentdb`.`content_genre`, CONSTRAINT `content_genre_ibfk_1` FOREIGN KEY

('content_id') REFERENCES `content` ('content_id')) 0.000 sec

Fix:

Used:

DROP TRIGGER IF EXISTS trg watchlist limit;

before every trigger. Also checked for stray text or copy-paste issues.

3. Table Already Exists Warnings

Error Message:

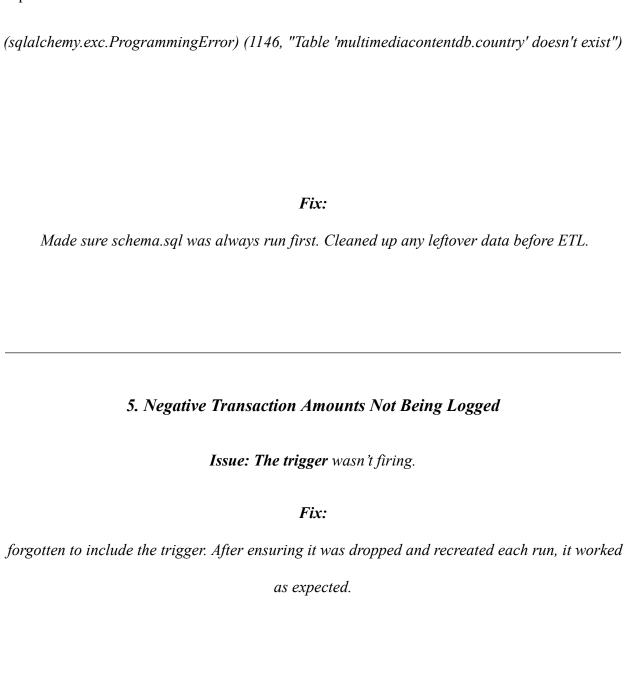
1050 Table 'error_log' already exists

Fix:

Ensured DROP TABLE IF EXISTS before each CREATE TABLE in the schema script.

4. ETL Load Errors

Error Message:



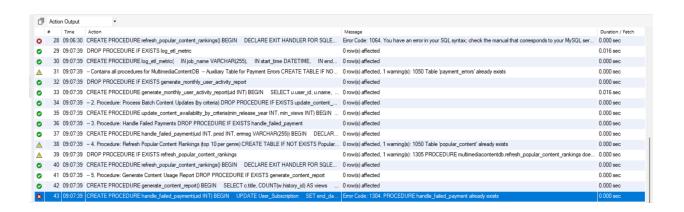
6. Scheduled Events Not Firing

Fix:

Enabled the event scheduler with

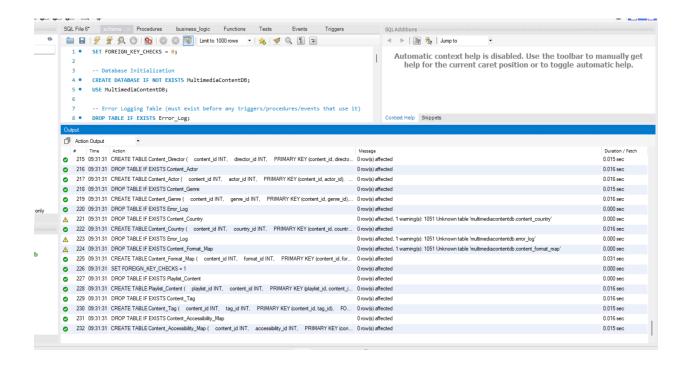
$SET\ GLOBAL\ event\ scheduler=ON;$

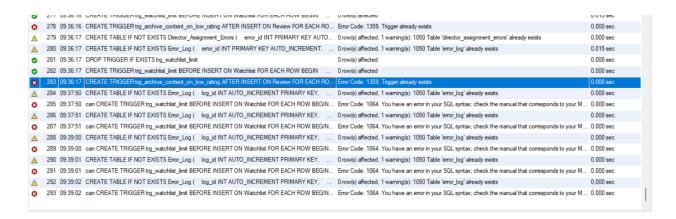
Then tested logic manually and checked Error_Log for entries.



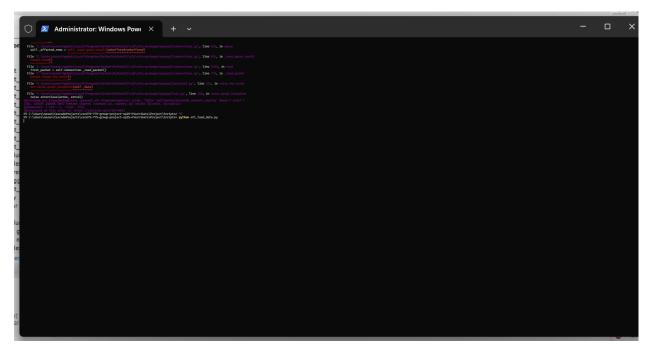
		Time	Action	Message	Duration / Fetch
•	35	09:07:39	CREATE PROCEDURE update_content_availability_by_criteria(min_release_year INT, min_views INT) BEGIN	0 row(s) affected	0.000 sec
0	36	09:07:39	3. Procedure: Handle Failed Payments DROP PROCEDURE IF EXISTS handle_failed_payment	0 row(s) affected	0.000 sec
•	37	09:07:39	${\sf CREATE\ PROCEDURE\ handle_failed_payment (uid\ INT,\ pmid\ INT,\ emmsg\ VARCHAR (255))\ BEGIN DECLAR}$	0 row(s) affected	0.000 sec
Δ	38	09:07:39	4. Procedure: Refresh Popular Content Rankings (top 10 per genre) CREATE TABLE IF NOT EXISTS Popular	0 row(s) affected, 1 warning(s): 1050 Table 'popular_content' already exists	0.000 sec
Δ	39	09:07:39	DROP PROCEDURE IF EXISTS refresh_popular_content_rankings	$0\ row(s)\ affected,\ 1\ warming(s):\ 1305\ PROCEDURE\ multimedia content db.refresh_popular_content_rankings\ doe$	0.000 sec
0	40	09:07:39	CREATE PROCEDURE refresh_popular_content_rankings() BEGIN DECLARE EXIT HANDLER FOR SQLE	0 row(s) affected	0.000 sec
•	41	09:07:39	5. Procedure: Generate Content Usage Report DROP PROCEDURE IF EXISTS generate_content_report	0 row(s) affected	0.000 sec
0	42	09:07:39	CREATE PROCEDURE generate_content_report() BEGIN SELECT c.title, COUNT(w.history_id) AS views	0 row(s) affected	0.000 sec
8	43	09:07:39	${\sf CREATE\ PROCEDURE\ handle_failed_payment(uid\ INT)\ BEGIN UPDATE\ User_Subscription SET\ end_da}$	Error Code: 1304. PROCEDURE handle_failed_payment already exists	0.000 sec
•	44	09:08:54	DROP PROCEDURE IF EXISTS log_etl_metric	0 row(s) affected	0.000 sec
•	45	09:08:54	${\sf CREATE\ PROCEDURE\ log_etl_metric} ({\sf\ IN\ job_name\ VARCHAR} (255), {\sf\ IN\ start_time\ DATETIME}, {\sf\ IN\ end}$	0 row(s) affected	0.000 sec
A	46	09:08:54	Contains all procedures for MultimediaContentDB Auxiliary Table for Payment Errors CREATE TABLE IF NO	0 row(s) affected, 1 warning(s): 1050 Table 'payment_errors' already exists	0.016 sec
•	47	09:08:54	DROP PROCEDURE IF EXISTS generate_monthly_user_activity_report	0 row(s) affected	0.000 sec
0	48	09:08:54	${\sf CREATE\ PROCEDURE\ generate_monthly_user_activity_report(uid\ INT)\ BEGIN SELECT\ u.user_id,\ u.name,\ \dots}$	0 row(s) affected	0.000 sec
•	49	09:08:54	$-2.\ Procedure: Process\ Batch\ Content\ Updates\ (by\ criteria)\ DROP\ PROCEDURE\ IF\ EXISTS\ update_content\$	0 row(s) affected	0.000 sec
0	50	09:08:54	${\sf CREATE\ PROCEDURE\ update_content_availability_by_criteria(min_release_year\ INT, min_views\ INT)\ BEGIN\ \dots}$	0 row(s) affected	0.000 sec
•	51	09:08:54	- 3. Procedure: Handle Failed Payments DROP PROCEDURE IF EXISTS handle_failed_payment	0 row(s) affected	0.000 sec
0	52	09:08:54	CREATE PROCEDURE handle_failed_payment(uid INT, pmid INT, emmsg VARCHAR(255)) BEGIN DECLAR	0 row(s) affected	0.016 sec
Δ	53	09:08:54	$-4. \ Procedure: Refresh Popular Content \ Rankings \ (top \ 10 \ per \ genre) \ CREATE \ TABLE \ IF \ NOT \ EXISTS \ Popular$	0 row(s) affected, 1 warning(s): 1050 Table 'popular_content' already exists	0.000 sec
0	54	09:08:54	DROP PROCEDURE IF EXISTS refresh_popular_content_rankings	0 row(s) affected	0.000 sec
•	55	09:08:54	CREATE PROCEDURE refresh_popular_content_rankings() BEGIN	0 row(s) affected	0.000 sec
0	56	09:08:54	5. Procedure: Generate Content Usage Report DROP PROCEDURE IF EXISTS generate_content_report	0 row(s) affected	0.000 sec
0	57	09:08:54	${\sf CREATE\ PROCEDURE\ generate_content_report()\ BEGIN} {\sf SELECT\ c.title,\ COUNT(w.history_id)\ AS\ views} \dots$	0 row(s) affected	0.000 sec
0	58	09:08:54	6. Procedure: Handle Failed Payment DROP PROCEDURE IF EXISTS handle_failed_payment	0 row(s) affected	0.015 sec
•	59	09:08:54	${\tt CREATE\ PROCEDURE\ handle_failed_payment(uid\ INT)\ BEGIN UPDATE\ User_Subscription SET\ end_da}$	0 row(s) affected	0.000 sec

```
SHOW CREATE PROCEDURE generate_monthly_user_activity_report;
10 -- 1. Procedure: Generate Monthly User Activity Report
11 DROP PROCEDURE IF EXISTS generate_monthly_user_activity_report;
12 CREATE PROCEDURE generate_monthly_user_activity_report(uid INT)
13 BEGIN
14
        SELECT u.user_id, u.name,
            COUNT(DISTINCT w.content_id) AS content_watched,
            AVG(r.score) AS avg_rating,
            SUM(TIMESTAMPDIFF(HOUR, w.watch_date, w.watch_date)) AS hours_spent
        FROM User u
18
                                                                                date >= DATE SUB(NOW
                                                                                     Save
                                                                                            Run ▼
    Procedure
                                    sql_mode
     generate_monthly_user_activity_report
                                     IGNORE_SPACE,ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,N
```









```
[parameters: {'cid': 1, 'ccid': 37}]

[Background on this error at: https://sqlalcho.me/e/78/f485)

[Sackground on this error at: https://sqlalcho.me/e/78/f485)

[PS C:\Users\nascen\CascadeProjects\csc675-775-group-project-sp25-Pierrikata\Project\Scripts> *C

PS C:\Users\nascen\CascadeProjects\csc675-775-group-project-sp25-Pierrikata\Project\Scripts> python etl_load_data.py

ETL complete!

PS C:\Users\nascen\CascadeProjects\csc675-775-group-project-sp25-Pierrikata\Project\Scripts> |
```

