TASK-3 DATABASE MIGRATION

Introduction

Migrating a database from one platform to another, such as from MySQL to PostgreSQL, is a critical process that requires careful planning, execution, and validation to ensure data consistency and integrity. MySQL and PostgreSQL are both robust relational database systems, but they differ in their syntax, data types, and capabilities. This guide provides a comprehensive step-by-step approach to migrate data efficiently while minimizing downtime and avoiding data loss.

Why Migrate from MySQL to PostgreSQL?

1. Advanced Features: PostgreSQL supports features like advanced indexing, window functions, and custom data types that are not natively available in MySQL.

2. Data Integrity: PostgreSQL has stronger support for ACID compliance and data validation.

3. Open Source Flexibility: PostgreSQL provides greater flexibility in managing extensions and custom functionalities.

4. Performance: For analytical and complex queries, PostgreSQL often outperforms MySQL.

Objectives of the Migration

1. Ensure schema compatibility between MySQL and PostgreSQL.

2. Safely transfer data while maintaining data integrity.

3. Validate the migrated data and ensure it aligns with business requirements.

4. Update the application to seamlessly connect to the PostgreSQL database.

Scope of the Guide

This guide covers the following key areas:

1. Preparation: Setting up the environment and analyzing schema differences.

2. Schema Conversion: Adapting MySQL schema to PostgreSQL.

3. Data Migration: Exporting data from MySQL and importing it into PostgreSQL.

4. Verification: Validating data integrity and application compatibility.

5. Summary Report: Documenting the process and outcomes.

Benefits of Following This Guide

1. A structured approach minimizes errors and reduces downtime.

2. Ensures a smooth transition without compromising data quality.

3. Provides reusable scripts and configurations for future migrations.

4. Helps teams align on best practices and compliance requirements.

Coding

#BACKUP MYSQL DATABASE

Command:

```
mysqldump -u [username] -p [database_name] > mysql_backup.sql
```

#SCHEMA CONVERSION

MySQL Schema (Original):

```
CREATE TABLE employees (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(255),
   active TINYINT(1),
   created_at DATETIME
);
```

Converted PostgreSQL Schema:

```
CREATE TABLE employees (
   id SERIAL PRIMARY KEY,
   name VARCHAR(255),
   active BOOLEAN,
   created_at TIMESTAMP
);
```

#DATA EXPORT

Export MySQL Data to CSV:

```
SELECT * INTO OUTFILE '/tmp/employees.csv'
```

```
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM employees;
```

Export Data (SQL Script):

```
mysqldump -u [username] -p --no-create-info [database_name] >
mysql_data.sql
```

#DATA IMPORT

Import Data into PostgreSQL Using CSV:

```
COPY employees (id, name, active, created_at)
FROM '/tmp/employees.csv'
DELIMITER ','
CSV HEADER;
```

Using pgloader (Configuration Script):

Save this configuration as migration.load:

```
LOAD DATABASE

    FROM mysql://[username]:[password]@localhost/[mysql_database]

    INTO
postgresql://[username]:[password]@localhost/[postgres_database]


WITH include no drop, create tables, create indexes, reset sequences


SET work_mem to '16MB', maintenance_work_mem to '512MB'


CAST type tinyint when (= precision 1) to boolean using (if zero then 'f'
else 't')

CAST type datetime to timestamp;


BEFORE LOAD DO

$$ create schema if not exists public; $$;
```

Run the command:

```
pgloader migration.load
```

#VERIFICATION

Row Count Check:

On MySQL:

```
SELECT COUNT(*) FROM employees;
```

On PostgreSQL:

```
SELECT COUNT(*) FROM employees;
```

Random Record Check:

```
SELECT * FROM employees LIMIT 10;
```

Null Value Check:

```
SELECT * FROM employees WHERE active IS NULL;
```

#APPLICATION CONFIGURATION

Update the database connection in your application. Example (PostgreSQL):

```
{
  "database": {
    "host": "localhost",
    "port": 5432,
    "user": "[username]",
    "password": "[password]",
    "database": "[postgres_database]"
  }
}
```

#DELIVERABLES

1. Scripts:

mysql_backup.sql (MySQL backup).

PostgreSQL schema script.

Data migration scripts (pgloader or CSV import).

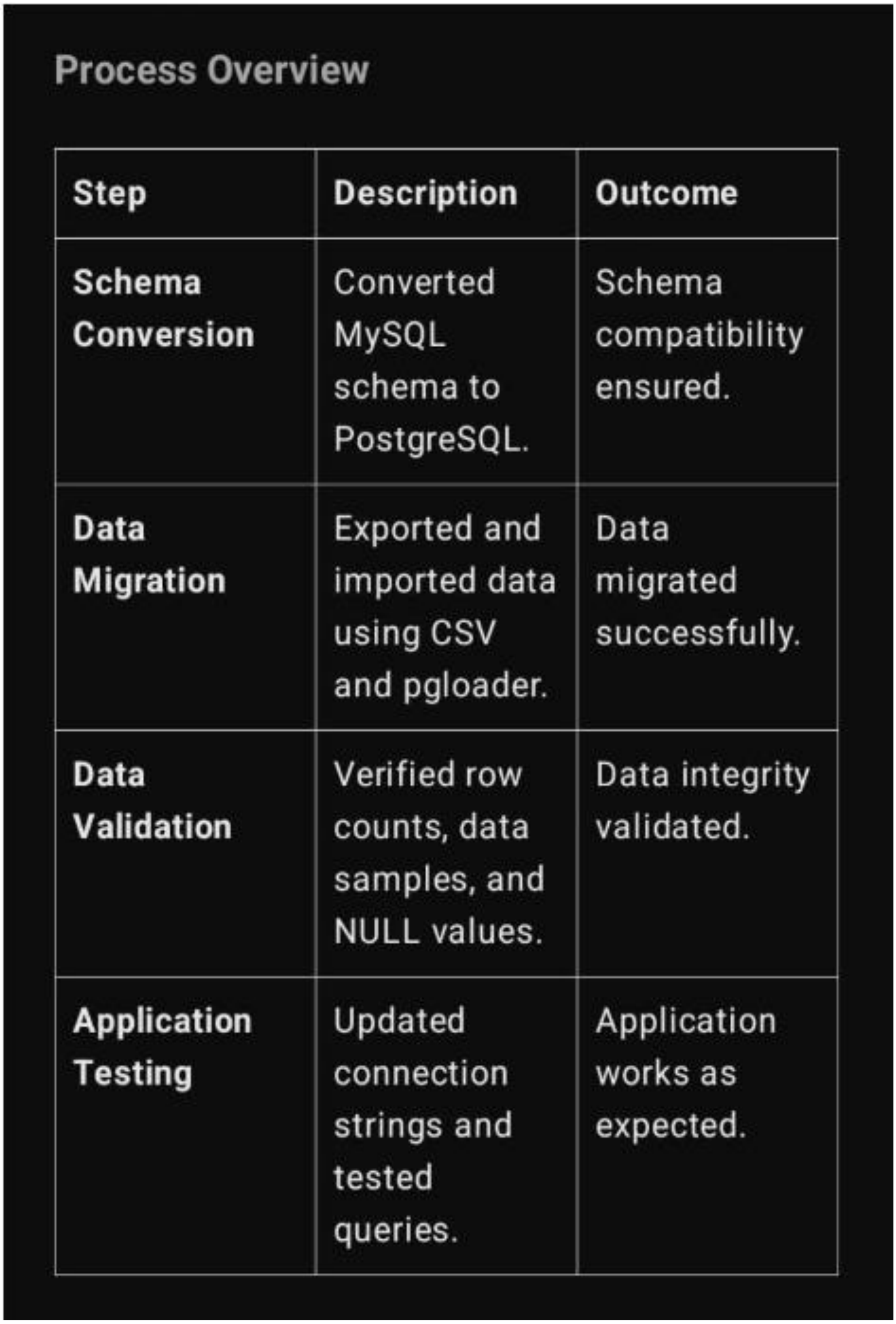
2. Report Template:

Row counts.

Examples of random record checks.

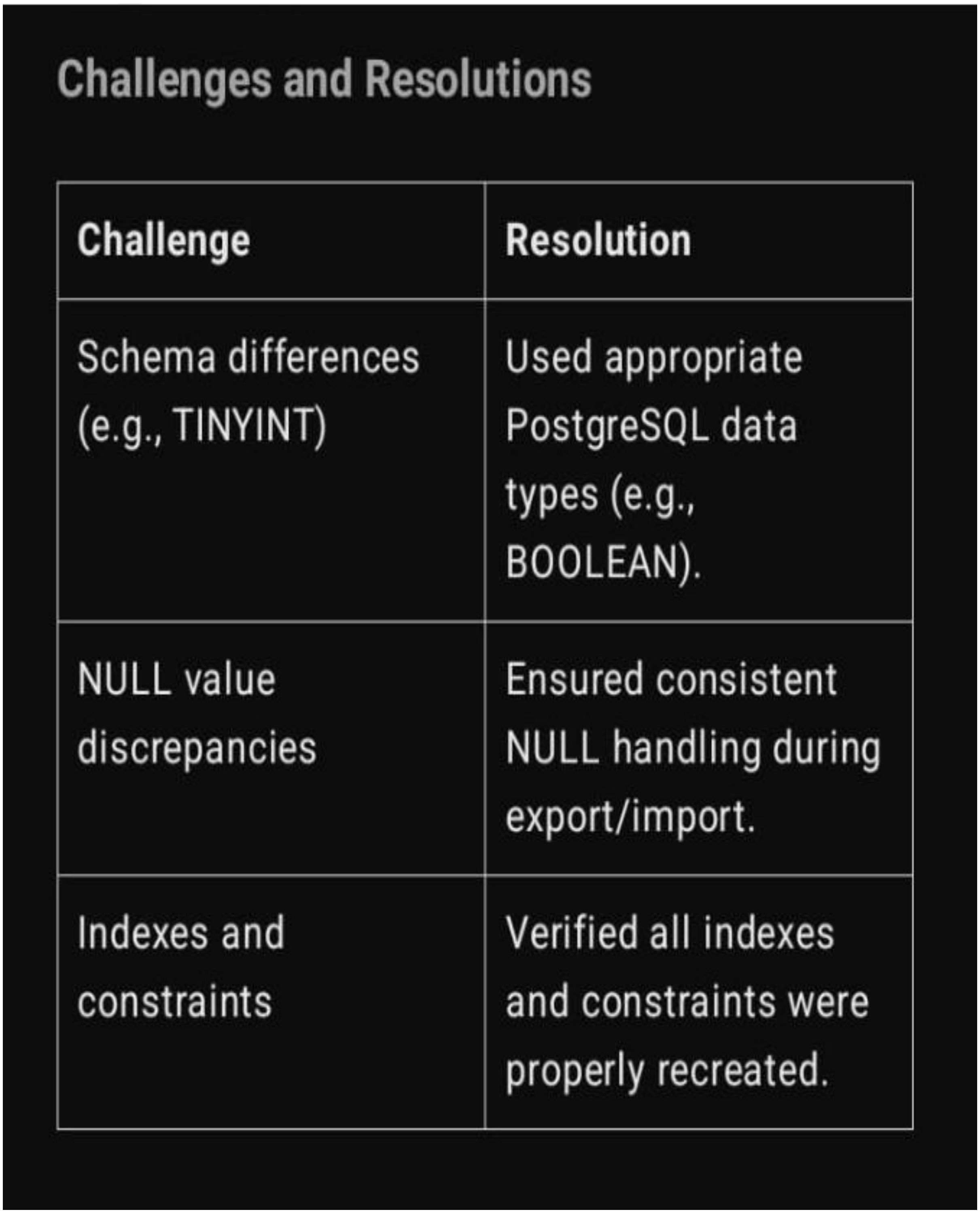
Any challenges and resolutions encountered.

Output



Process Overview

Step	Description	Outcome
Schema Conversion	Converted MySQL schema to PostgreSQL.	Schema compatibility ensured.
Data Migration	Exported and imported data using CSV and pgloader.	Data migrated successfully.
Data Validation	Verified row counts, data samples, and NULL values.	Data integrity validated.
Application Testing	Updated connection strings and tested queries.	Application works as expected.



Challenges and Resolutions

Challenge	Resolution
Schema differences (e.g., TINYINT)	Used appropriate PostgreSQL data types (e.g., BOOLEAN).
NULL value discrepancies	Ensured consistent NULL handling during export/import.
Indexes and constraints	Verified all indexes and constraints were properly recreated.