# Data Model – Test Data Integration ERD

This document describes the proposed normalized data model for unified test data ingestion and reporting across Jira, GitLab, and JMeter. The model provides a structured representation of entities, relationships, and data sources used in the test lifecycle from platform configuration through execution tracking.

## 1. Overview

The following logical data model supports integration between delivery and testing systems. It ensures referential integrity, normalized structure, and clear traceability between configuration and execution layers.

## 2. Proposed Entities and Relationships

|  |  |  |  |
| --- | --- | --- | --- |
| Entity | Key Fields | Relationships | Source System |
| Platform | platform\_id, platform\_name | 1:N → AgileReleaseTrain | JMeter |
| AgileReleaseTrain | art\_id, art\_name, platform\_id | 1:N → Project | JMeter |
| Project | project\_id, project\_name, art\_id | 1:N → Release | JMeter |
| Release | release\_id, release\_name, project\_id | 1:N → TestCycle; N:M → System | Jira |
| System | system\_id, system\_name | M:N → Release | JMeter |
| Release\_System\_Map | release\_id, system\_id, system\_role | Join table for Source/Target mapping | Derived |
| TestCycle | cycle\_id, cycle\_name, release\_id | 1:N → TestRun | Jira |
| TestRun | test\_run\_id, test\_run\_name, cycle\_id | 1:N → Test | GitLab |
| Test | test\_id, test\_run\_id | Leaf entity | Jira / GitLab |

## 3. Detailed Field Definitions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity | Field | Type | Key | Description |
| Platform | platform\_id | int | PK | Unique platform identifier |
| Platform | platform\_name | varchar(100) |  | Name of the platform |
| AgileReleaseTrain | art\_id | int | PK | Unique ART identifier |
| AgileReleaseTrain | art\_name | varchar(100) |  | Agile Release Train name |
| AgileReleaseTrain | platform\_id | int | FK → Platform.platform\_id | Parent platform |
| Project | project\_id | int | PK | Unique project identifier |
| Project | project\_name | varchar(100) |  | Project name |
| Project | art\_id | int | FK → AgileReleaseTrain.art\_id | Owning Agile Release Train |
| Release | release\_id | int | PK | Unique release identifier |
| Release | release\_name | varchar(100) |  | Release name or version label |
| Release | project\_id | int | FK → Project.project\_id | Owning project |
| System | system\_id | int | PK | Unique system identifier |
| System | system\_name | varchar(100) |  | Application or system name |
| Release\_System\_Map | release\_id | int | FK → Release.release\_id | Associated release |
| Release\_System\_Map | system\_id | int | FK → System.system\_id | Associated system |
| Release\_System\_Map | system\_role | varchar(50) |  | Role of system in flow (e.g., source/target) |
| TestCycle | cycle\_id | int | PK | Unique test cycle identifier |
| TestCycle | cycle\_name | varchar(100) |  | Named cycle or suite |
| TestCycle | release\_id | int | FK → Release.release\_id | Associated release |
| TestRun | test\_run\_id | int | PK | Unique test run identifier |
| TestRun | test\_run\_name | varchar(100) |  | Test run label |
| TestRun | cycle\_id | int | FK → TestCycle.cycle\_id | Owning test cycle |
| Test | test\_id | int | PK | Unique test identifier |
| Test | test\_run\_id | int | FK → TestRun.test\_run\_id | Owning test run |

## 4. Normalization and Design Notes

- Each entity includes a unique primary key (\_id) field.  
- Relationships are defined explicitly using foreign keys.  
- Release\_System\_Map normalizes the many-to-many mapping between Releases and Systems and supports system\_role to distinguish Source/Target.  
- Naming conventions follow lowercase snake\_case aligned with PostgreSQL standards.  
- Data types:  
 - int for identifiers  
 - varchar for descriptive attributes (lengths may vary by field)  
- The model is designed for future enrichment (e.g., test result metrics).

## 5. Clarification Questions

1. Is the hierarchy Platform → AgileReleaseTrain → Project → Release always consistent, or can projects span multiple ARTs?
2. Should source and target systems be represented within a single mapping table (Release\_System\_Map) or separate entities?
3. Can a single release have multiple source and multiple target systems concurrently?
4. Are test\_run\_id and test\_id globally unique across Jira and GitLab?
5. What is the preferred ingestion method — direct API integration, JSON intermediary, or CSV extract?
6. Should IDs be system-generated locally or aligned with upstream Jira/GitLab identifiers?
7. Are lowercase snake\_case naming conventions confirmed for the PostgreSQL schema?
8. Is historical version tracking required, or will only current snapshots be maintained?

## 6. Next Steps

- Confirm responses to clarification questions.  
- Finalize PostgreSQL schema DDL and constraints.  
- Map upstream data fields from Jira, GitLab, and JMeter APIs to the target entities.  
- Validate relationships and populate initial reference data.  
- Review and approve for integration with the data ingestion pipeline.

## 7. Entity–Relationship Diagram (ERD)

The final ERD produced at the conclusion of this exercise is shown below:

