Database Design Technical Design Document

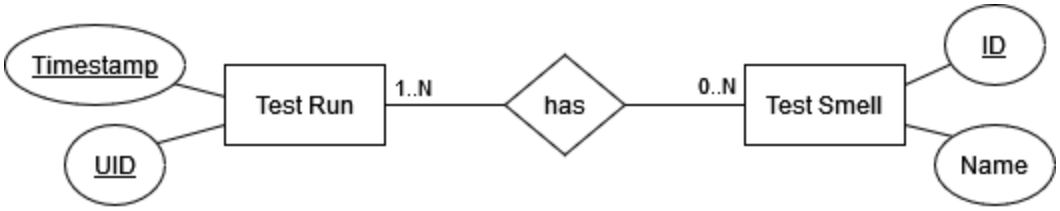
MySQL

The decision to choose MySQL as our database management system was fueled by several factors. The first was that MySQL is offered as an open source database solution with open source approved license (GNU), which aligns with the TSDetect tool also being open source, enabling users of TSDetect to do as they wish with both the tool and their results. A paid option would be an infringement on the TSDetect ethos, as such a free open source option is the ideal choice. Another factor that played into this decision was the compatibility that MySQL has with all major operating systems (Windows, MacOS, Linux, etc.), which also adheres to our open source standards and allows developers to make use of TSDetect without being constrained by their operating system. A third factor is the fact that MySQL is structured as a relational database system, thus allowing us to have a schema that will allow for the leveraging of multiple inter-related tables which in turn will more easily permit expansion should new test smells be added to the tool. The final factor that played into this decision is the row limit imposed on other databases. The database limit that MySQL has of 256 TB is such a value that given our comparably small amount of data should not be a restricting factor in our development

Necessary Data

The data we currently want to store is the smell type and their associated counts from the results of the tool. Each user will have a unique identifier (UID) and in order to differentiate the runs for a single user, we need to store the timestamp of the run as well. Users are not likely to detect every smell type on every run of the tool, so we only want to store non-zero counts for a smell. This will help to save space in the database as well. The data we are storing from users is completely anonymous and contains no personal information.

Entity Relationship Diagram



We had originally proposed a single table schema that was as follows:

test_runs(uid, timestamp, smelly_test_count, lazy_test_count, etc..)

Although the initial design stores identical information and utilizes the same primary keys, adding new test smells becomes impossible without adding new columns to the table.

At minimum, the team determined that two entities would be required: test run & test smell. The relationship between the the two entities exists for the following reasons:

- A single test run must exist for test smells to exist. This results in the participation for this relationship being partial since a test run entity may exist without a given test smell being discovered.
- The cardinality of the relationship is many-to-many as the test run entity may have multiple test smells that arise (ex: lazy tests, slow tests, etc.). As for test smells, a single test smell may be found in multiple test runs of various projects.

Reduction to Tables

test_runs(uid, timestamp, run_id)
test_smells(test_smell_id, name)
test_run_smells(run_id, *test_smell_id*, quantity)

The above Entity relationship Diagram translates intuitively to a set of three tables:

1. test_runs, which contains the “uid” of the account that ran the test, paired with the “timestamp” of the run. The uid and the timestamp will form a composite key on this table. A unique run_id will also be generated when the test run is added to the table, and will be utilized as a foreign key in the test_run_smells table as detailed below. The uid takes the form of a CHAR of size (X), and the timestamp is a DATETIME entry, in the

UTC time zone, and the run_id will be a BIGINT, to maximize the possible number of entries.

2. test_smells will be a table identifying the test smell types (detailed here). Each Test smell type has two components: a unique identifying number called “test_smell_id”, and the test “name”. The test_id is a TINYINT generated at the time the test is first entered into the table, and the test_id is a TINYTEXT, containing the English name for the test smell.

3. test_run_smells is a table representing the many-to-many “has” relationship between the test_runs table and the test_smells table. It contains run_id, associated with a test_id and a “quantity”. The run_id and the test_smell_id will form a composite key, associating a given test run with the quantity of the given test smell that appeared in that run. The run_id and test_smell_id will share the data types of their counterparts in the first two tables, and the “quantity” will be a BIGINT. “Quantity” should never have a zero value; in the event that a test smell does not appear in a given test run, the run_id - test_smell_id pairing will not appear in the table.

The intention behind this breakdown is three fold: first, it allows for new test smells to be introduced into the database without the need to alter the existing columns or duplicate tables. Secondarily, this reduces the maximum possible row size, which is the most restrictive size-related element in MySQL. Third, using this structure means that if new users want to collect different data from the user, this data can be added to the database while imposing the least possible number of restrictions.