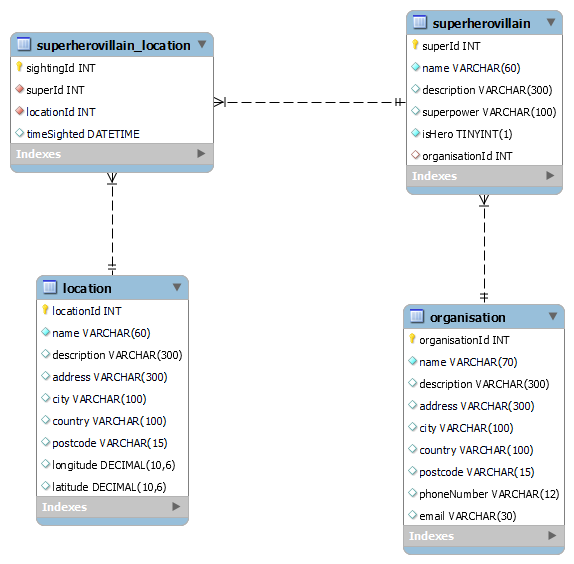
Superhero Sightings Part 1: Data Layer

# README

To run this file, firstly run the superheroes set up, and superheroes test set up SQL files. There is no real functionality when running the application yet, however the tests can all be run. There are 4 test files to test the 4 entities.

# ERD Diagram



Some assumptions made:

* Organisation and Superherovillain has a one-to-many relationship, i.e. a superhero can only be part of one organisation but an organisation can consist of many relationships
* Location and superherovillains have a many-to-many relationship, but as the superherovillain\_location bridge table is treated as an entity for the sightings, it has its own primary key to make it easier to work with (i.e. finding by id)
* Initially I had planned for superhero and supervillain be different tables but since they would essentially have the exact same information, they were combined with an isHero Boolean method instead to avoid redundancy

# Requirements

|  |  |
| --- | --- |
| **ERD - Second Normal Form: The database design meets AT LEAST the requirements for second normal form.** | Done |
| **ERD - Primary Keys: Each table has an appropriate primary key named either id or entityId or an appropriate composite key with all fields named correctly.** | Done – used entityId specifically |
| **ERD - Relationships: The relationships between the tables are expressed correctly with foreign keys on entity tables for one-to-many relationships and bridge tables for many-to-many relationships.** | Done (see ERD)  One-to-many with organisationid within the superhero table  Many-to-many with the hero\_location bridge table |
| **ERD - Data Columns: Each field has reasonable types and scales.** | Done |
| **Script Re-runnable: The script used to create the database can be re-run as needed, including dropping the database, rebuilding it, and USE the database.** | Done |
| **Script - Fits ERD Specification: The script creates tables named as presented in the ERD, with columns specified according to the ERD requirements, all foreign key relationships present, and auto-incrementing where appropriate.** | Done |
| **DTOs - Full Entity Modeling: All database entities have equivalent class representations with appropriate fields, getters, and setters.** | Done  Classes: Superherovillain, Sighting, Location, Organisation |
| **DAOs - Separation of Concerns: Each entity has its own DAO.** | Done  Classes: SuperherovillainDao, SightingDao, LocationDao, OrganisationDao |
| **DAOs - Interface Extraction: Each DAO implements a method that defines all of its public methods.** | Done  Classes: SuperherovillainDaoDB, SightingDaoDB, LocationDaoDB, OrganisationDaoDB |
| **DAOs - Full CRUD Operations: Each DAO implements add, getById, getAll, edit, and delete methods.** | Done |
| **Unit Testing: Golden Path Testing Unit tests verify that all public dao methods perform their operations as expected given valid inputs, including validation persistence checks that do not trust the return value of the method being tested.** | Done |
| **Unit Testing -Bad Input Testing: Unit tests verify that all public dao methods produce the correct exception types on invalid inputs, including invalid ids and nulls.** | Done |
| **Code Style: All code is written in a readable, conventions-compliant manner.** | Done |