# Database Design for WESH pilot plant

Bishnu Poudel
MS in Data Science: 2019-21
Norwegian University of Life Sciences
Course: INF230 - Data Processing and Analysis

**Abstract**

The report contains the following in relation to the fictional research group Water Sanitation Environment and Health WESH, that wants to modernize their data analysis/gathering process for their pilot plant.

1) A description and evaluation of the modelling process where I explain the choices I make.
2) Assessment of each table and their relations
   a) How do they fulfil the requirements set?
   b) Potential problems and considerations when expanding the model in the future
      i) What if operators need to register if they are qualified to work in the lab/plant?
3) An image of the final model
4) The necessary queries.

## I. Modeling Method

### A. *Reverse Engineering MySQL - Workbench*

First the three given csv files were loaded into a different MySQL datbase by importing them IMPORT_EXISTING_CSV.sql. Then using the reverse engineering option on Workbench, these tables were imported into a model. I used these tables as the starting point to create the database design. Then I created the following tables in the models in the given order.

- operators_details
- test_group
- test_description
- samples_taken
- analysis_done
- BarCode_Gen

### B. *Forward Engineering Workbench - MySQL*

After the modeling was complete, using the forward engineering option the CREATE_SCRIPT.sql was automatically generated. I created the tables in a new database using this scripts. A trigger was used on the samples_taken table to generate the bar code.

### C. *Potential Insert scripts when the system is implemented*

A number of useful queries are available on the INSERT_SCRIPTS.sql for potential use when the project gets implemented.

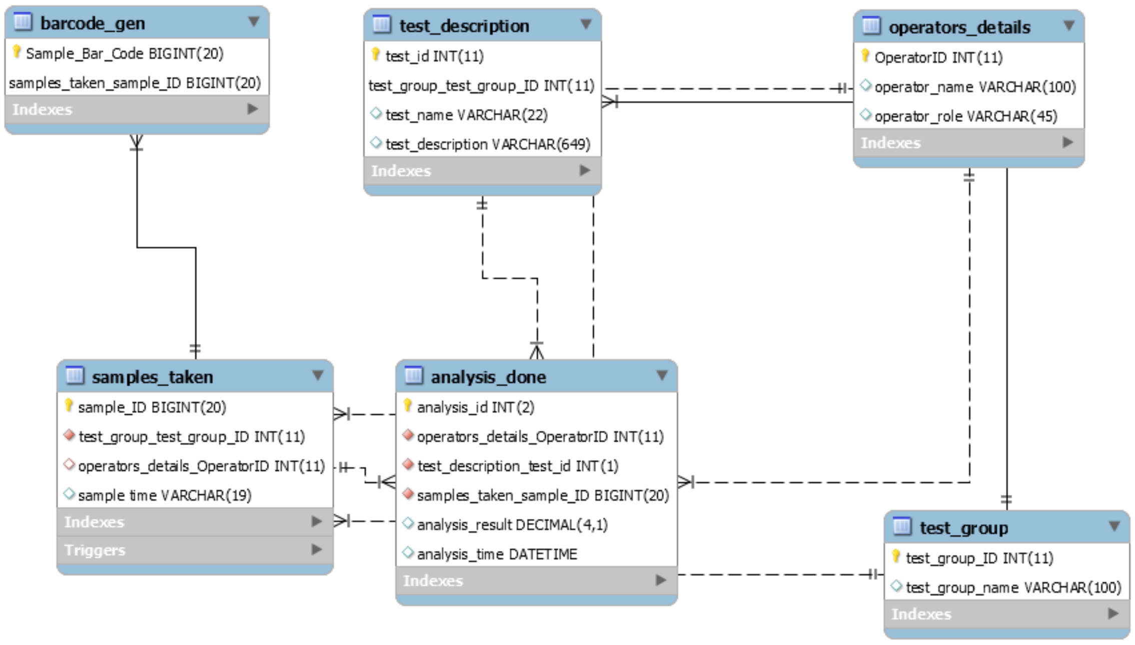
## II. Breakdown of how each table fulfills the requirements

We can change the non-identifying relations between the tables into an identifying relationship if we want to make the foreign key relationship more restrictive.

### A. *operators_details, test_group and test_description*

These tables are static tables, with details about tests and operators, and with an auto-incremented ID as primary key. There are no transitive or partial dependencies, so the tables are in third normal form.

Operator details table is linked with both the samples taken and analysis done table on a one to many basis.



barcode_gen
Sample_Bar_Code BIGINT(20)
samples_taken_sample_ID BIGINT(20)
Indexes
test_description
test_id INT(11)
test_group_test_group_ID INT(11)
test_name VARCHAR(22)
test_description VARCHAR(649)
Indexes
operators_details
OperatorID INT(11)
operator_name VARCHAR(100)
operator_role VARCHAR(45)
Indexes
samples_taken
sample_ID BIGINT(20)
test_group_test_group_ID INT(11)
operators_details_OperatorID INT(11)
sample time VARCHAR(19)
Indexes
Triggers
analysis_done
analysis_id INT(2)
operators_details_OperatorID INT(11)
test_description_test_id INT(1)
samples_taken_sample_ID BIGINT(20)
analysis_result DECIMAL(4,1)
analysis_time DATETIME
Indexes
test_group
test_group_ID INT(11)
test_group_name VARCHAR(100)
Indexes

Fig. 1. WESH database model

### B. *samples_taken and BarCode_Gen*

Any new sample taken must have a test group, operator ID, and the timestamp when the sample was taken. The table is in third normal form. Whenever a new entry is made into the samples_taken table, a trigger is run on BarCode_Gen to generate a unique bar code for that sample.

### C. *analysis_done*

This table is perhaps the most frequently used table. It has links to all the other tables. Therefore, all information can be extracted by joining with other tables. It has many to one relation with 3 other tables.

## III. Image of the Final Model

The figure for the database model is figure 1.

## IV. Queries for Using Database

### A. *Insert into samples_taken*

Every time a new sample is taken, BarCode_Gen is populated as a result of the trigger. The variables @test_group_name, @operator_name come as input from the website/app.

```
SELECT  @test_group_id  :=  test_group_id
FROM    test_group
WHERE   Lower(test_group_name) = Lower(@test_group_name);

SELECT  @operator_id  :=  operatorid
FROM    operators_details
WHERE   Lower(operator_name) = Lower(@operator_name);

INSERT INTO  `wesh`.`samples_taken`
```

```
                 (test_group_test_group_id ,
                  operators_details_operatorid ,
                  'sample time ')
SELECT @test_group_id ,
       @operator_id ,
       CURRENT_TIMESTAMP();
```

### *B. Insert into analysis_done*

Inserts a new row into analysis done, for each new test. The variables @operator_name, @Bar_Code come as input from the website/app.

```
SELECT @operators_details_operatorid := operators_details_operatorid
FROM   operators_details
WHERE  Lower(operator_name) = Lower(@operator_name);

CREATE temporary TABLE IF NOT EXISTS test_ids_table AS
(SELECT DISTINCT d.test_id AS test_id
FROM   test_description AS d
INNER JOIN (SELECT a.*,
              b.*
FROM   barcode_gen AS a
              INNER JOIN samples_taken AS b
                          ON a.samples_taken_sample_id = b.sample_id
WHERE  Lower(sample_bar_code) = Lower(@bar_code)) AS c
ON c.test_group_test_group_id = d.test_group_test_group_id);

SELECT @sample_id := samples_taken_sample_id
FROM   barcode_gen
WHERE  Lower(sample_bar_code) = Lower(@bar_code);

INSERT INTO wesh.analysis_done
(operators_details_operatorid ,
test_description_test_id ,
samples_taken_sample_id ,
analysis_result ,
analysis_time)
SELECT @operators_details_operatorid ,
test_id ,
@sample_id ,
@analysis_result ,
CURRENT_TIMESTAMP()
FROM   test_ids_table;
```

## V. Discussion