Coding Exercise

Overview

This exercise will assess code organization, robustness, database design, and problem solving.

To demonstrate this, you will be tasked with designing a database, write a reusable load script, and a report script.

When the exercise is complete, you must provide the assets to both run this code and standup any resources.

Documentation must be included. It should describe how to setup an environment to run this code. If non-standard/non-core utilities are used, they must be declared in the documentation.

You can assume in our environment that a database has already been created and that Python 3.7 and bash are present.

Technologies

Scripting language can be any combination of bash and python 3.7.

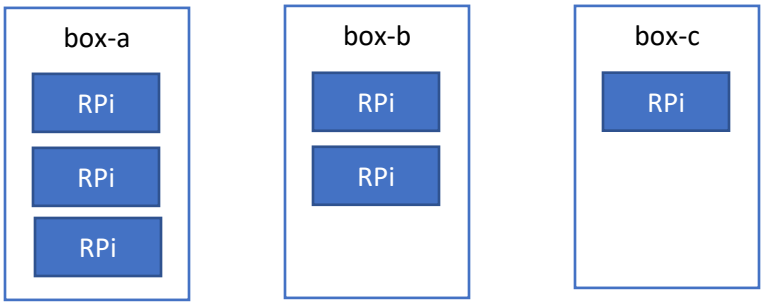
Database must be MySQL 5.7

Code will run on a Debian 10 host.

Exercise

You will design an inventory database, create load and report scripts for that database.

The inventory database will record data about boxes. These boxes can contain one or many RPis (Raspberry PIs).



Database Design

Each box is represented with the following properties:

- box name
- box build date

The box name will be unique.

A box also includes a collection of RPis. Each RPi is represented with the following properties:

- MAC address
- purchase date
- condition

The MAC address is unique across RPis.

Only one RPi can only be associated with one box at a time.

The design should also allow us to reallocate a RPi to another box. For this exercise, you are not expected to implement this mechanism.

Database must be designed for MySQL 5.7

When submitting you code, provide only the necessary SQL to setup the database. The data is not required.

Load Script

The data will be provided in a loose comma-separated values format. Each line in the file will include the following:

- box name
- box build date
- RPi MAC address
- RPi purchase date
- RPi condition

Each box will include at least one RPi. The MAC address, purchase date, and condition are distinct to each RPi.

The number of fields will vary depending on the number of RPis in a box (ex. 1 RPi = 5 fields, 2 RPis = 8 fields, 3 RPis = 11 fields, etc.).

The RPi condition can be one of the following: "unused", "good", "failing", "dead"

Sample CSV lines:

```
box-a, "2020-06-01 15:34:01", 00:0d:83:b1:c0:8f, "2020-01-30 14:00:00", "good",
00:0d:83:b1:c0:80, "2020-01-31 11:00:00", "unused", 00:0d:83:b1:c0:81, "2020-02-01
11:00:00", "good"
box-b, "2020-06-02 13:21:10", 00:0d:83:b1:d0:8f, "2020-01-30 14:00:00", "good",
00:0d:83:b1:d0:80, "2020-01-31 11:00:00", "good"
box-c, "2020-06-02 14:13:10", 00:0d:83:b1:e0:8f, "2020-01-30 14:00:00", "good"
```

You can assume that data file will never include a field header.

The import script should accept an argument for the data file. It is not necessary for the script to accept multiple datafiles, but it's also not discouraged.

Expected invocation:

```
$ box-data-import DATAFILE
```

Report Script

Report should list boxes where at least one RPi's purchase date is earlier than a provided date or between two provided dates.

For boxes which match the search criteria, the following should be printed:

- box name
- box build date
- for each RPi:
    - RPi MAC address
    - RPi purchase date
    - RPi condition

Output should be represented in JSON format. For example, if only "box-b" matched the search criteria, the script should return this:

```
{
    "box name": "box-b",
    "box build date": "2020-06-02 13:21:10",
    "RPi": [
        {
            "MAC address": "00:0d:83:b1:d0:8f",
            "purchase date": "2020-01-30 14:00:00",
            "condition": "good"
        },
        {
            "MAC address": "00:0d:83:b1:d0:80",
            "purchase date": "2020-01-31 11:00:00",
            "condition": "good"
        }
    ]
}
```

Expected invocation:

```
$ box-data-report OLDER-THAN-DATE [NEWER-THAN-DATE]
```