Database Design

## Purpose

This document outlines the structure of the SQL database for Assignment 4 of the MSc. Software Engineering Module. It describes the static and dynamic data imported in JSON format from [JCDecaux Developer](https://developer.jcdecaux.com/#/opendata/vls?page=static&contract=Dublin) and suggests a database structure for the project.

## Data Structure

**Static Data**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Data Type** | **Description** | **Comment** |
| number | **Int** | Number of the station. This is NOT an id, thus it is unique only inside a contract. | Without multiple contracts (i.e. multiple cities), this can become the unique identifier for the stations. |
| contract\_name | **Varchar** | Name of the contract of the station | Constant column. Only one city being analysed. Should be dropped. |
| name | **Varchar** | Name of the station | Looking at data, both the name and address have the same values. |
| address | **Varchar** | Address of the station. As it is raw data, sometimes it will be more of a comment than an address | Same as name |
| position | **Float** | position of the station in WGS84 format | Data is split into two: 1) longitude and 2) latitude. |
| banking | **Boolean** | Indicates whether this station has a payment terminal | Not included in Dublin static data. Not required. |
| bonus | **Boolean** | Indicates whether this is a bonus station | Not included in Dublin static data. Not required. |

**Dynamic Data**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Data Type** | **Description** | **Comment** |
| status | **Varchar** | Indicates whether this station is CLOSED or OPEN | - |
| bike\_stands | **Int** | The number of operational bike stands at this station | - |
| available\_bike\_stands | **Int** | The number of available bike stands at this station | - |
| available\_bikes | **Int** | The number of available and operational bikes at this station | - |
| last\_update | **Data/Time** | Timestamp indicating the last update time in milliseconds since Epoch | Should be in a format that can be used in line with the open weather application. |

With each instance of real-time/dynamic data, the static data is again replicated. See example below:

**Example of Real-time data**

{

"number": 123,

"contract\_name" : "Paris",

"name": "stations name",

"address": "address of the station",

"position": {

"lat": 48.862993,

"lng": 2.344294

},

"banking": true,

"bonus": false,

"status": "OPEN",

"bike\_stands": 20,

"available\_bike\_stands": 15,

"available\_bikes": 5,

"last\_update": <timestamp>

}

## Proposed Design

Two simple tables can be created for the database for 1) static and 2) dynamic. Since the dataset is only dealing with Dublin as a contract, all numbers are unique. The number should therefore be the primary key for static data. The only constant column: contract\_name will also be dropped as we’re only dealing with Dublin city data.

For dynamic data, we’re dealing with multiple weeks of data. As such, the primary key for the dynamic data will be the number and the last\_update column in conjunction. Both tables will link via the number as its foreign key. To reduce replication and address complications that arise from data repetition, the tables will be normalised. All data present in the dynamic data will not be scraped, except for the number which is required for the primary key.

See below the proposed design in an entity relationship diagram:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Static**  Foreign Key | |  | **Dynamic** | |
| PK | number : int (FK) |  | PK | number : int (FK) |
|  | name : varchar |  | last\_update : Time/Date |
|  | address:: varchar |  |  | status: varchar(7) |
|  | longitude: real |  |  | bike\_stands: int |
|  | latitude: : real |  |  | available\_bike\_stands : int |
|  |  |  |  | available\_bikes : int |

\* Real is a float in SQLite3

ER Diagram source: <http://www.datanamic.com/dezign/erdiagramtool.html>