**Appendix 12 – Database design**

The approach employed to design a database for the Downtime Management System will be like the one presented in ‘Relational Database Design and Implementation, 4th Edition’ by Jan L. Harrington (2016). The database for the Downtime Management System shall be relatively simple. The reason for taking a simplified approach is that the database will only support some of the functionalities of the prototype application and it is not a part of the final system. Therefore, the only a minimalistic database is required containing very few sample entities.

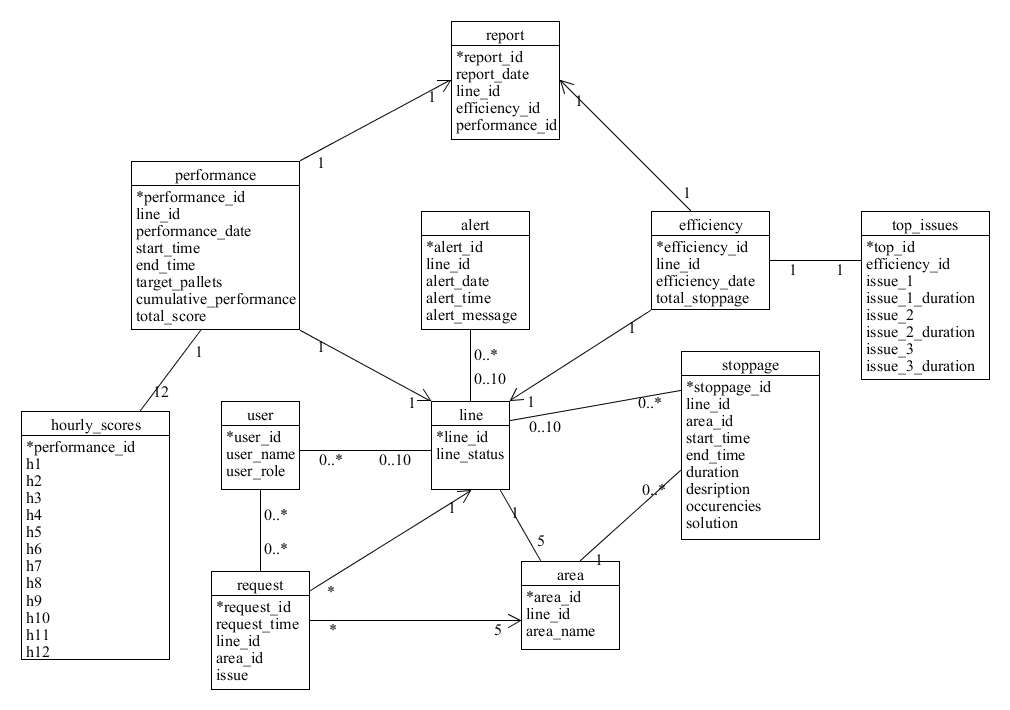
To complete the database design the following steps will be taken:

* Identifying entities
* Identifying attributes
* Identifying relationships
* Identifying domains

Much of the work has been accomplished in the previous design phases, that is, system design and user interface design. Previously created models will be now reviewed to identify entities for which the data needs to be stored.

Considering entities uncovered the need for grouping different types of users into one class for convenience as they share attributes and differ only in terms of permissions. Also, permissions do not need to be a separate attribute as they can be restricted by the user’s role. While the prototype is being built with the one organisation in mind and it will be consistent with their requirements it shall be easy to adapt to a different environment. Hardcoding roles wouldn’t be flexible enough to match this need as different role naming convections would require changes in data tables and application classes.

The figure below shows the database model consisting of entities with their attributes. Attributes marked with asterisks are the unique identifiers (primary keys) used to improve the overall performance of the database. Some database tables might include attributes not identified in the system and UI design stages. Relationships are represented using UML style.



**Figure 22** Database design model for the Downtime Management System.

To accomplish the database design, we need to associate attributes with domains, that is, data types allowed for each of them. It is important to note that for the development of the database Java DB will be used. Therefore, some of the data types might not be identical with those available in SQL. The table below shows the data types for each of the attributes. Attributes that are the same for two or more entities have been assigned data type only in the first occurrence.

**Table 15** Data types for entity attributes.

|  |  |  |
| --- | --- | --- |
| **Entity** | **Attribute** | **Domain** |
| user | user\_id | CHAR |
|  | user\_name | CHAR |
|  | user\_role | CHAR |
| line | line\_id | CHAR(2) |
|  | line\_status | BIT |
| area | area\_id | CHAR |
|  | area\_name | CHAR |
| stoppage | stoppage\_id | CHAR |
|  | start\_time | TIME |
|  | end\_time | TIME |
|  | duration | INTEGER |
|  | description | LONGVARCHAR |
|  | occurrences | INTEGER |
|  | solution | LONGVARCHAR |
| alert | alert\_id | CHAR |
|  | alert\_date | DATE |
|  | alert\_time | TIME |
|  | alert\_message | LONGVARCHAR |
| request | request\_id | CHAR |
|  | request\_time | TIME |
|  | issue | LONGVARCHAR |
| performance | performance\_id | CHAR |
|  | performance\_date | DATE |
|  | target\_pallets | INTEGER |
|  | cumulative\_performance | DECIMAL |
|  | total\_score | INTEGER |
| hourly\_scores | h1..h12 | INT(2) |
|  |  |  |
| efficiency | efficiency\_id | CHAR |
|  | efficiency\_date | DATE |
|  | total\_stoppage | INTEGER |
| top\_issues | top\_id | CHAR |
|  | issue\_1..3 | LONGVARCHAR |
|  | isssue\_1..3\_duration | INT |
| report | report\_id | CHAR |
|  | report\_date | DATE |