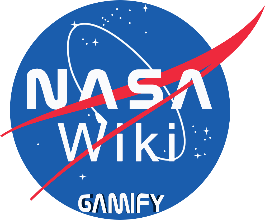
**NASA EVA**

**Gamification**

**D A T A B A S E D E S I G N**

**PHASE II**



**Prepared By**

Laura Addiego

Samia Alam

Kelli Corey

Charles Milk

Adeola Odusola

Hung Pham

# Table of Contents

[Table of Contents 2](#_Toc518220300)

[Revision Table 2](#_Toc518220301)

[Overview 3](#_Toc518220302)

[Specific Terms and Acronyms 3](#_Toc518220303)

[Hardware and Software 3](#_Toc518220304)

[Diagram Tool 3](#_Toc518220305)

[Database 3](#_Toc518220306)

[SQL Editor 3](#_Toc518220307)

[DDL and DML 3](#_Toc518220308)

[ERD 3](#_Toc518220309)

[Relationships 5](#_Toc518220310)

[Cardinalities/Business Rules 5](#_Toc518220311)

[Entities 5](#_Toc518220312)

[Assumptions and Special Considerations 8](#_Toc518220313)

[DDL Statements 8](#_Toc518220314)

# Revision Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Version Number** | **Description of Change** | **Author** | **Date** |
| 1.0 | Initial Creation of Document | Laura Addiego Prospero | 06/25/2018 |
|  |  |  |  |

# Overview

The database described in this document will augment the default MediaWiki database to allow for the gamification of the NASA EVA MediaWiki. The database will store the values necessary to track point earned by users, badge reward and leaderboard.

# Specific Terms and Acronyms

|  |  |
| --- | --- |
| **Term/Acronym** | **Description** |
| SQL | Structured Query Language |
| DDL | Data Definition Language |
| DML | Data Manipulation Language |
| ERD | Entity Relationship Diagram |

# Hardware and Software

## Diagram Tool

SQL Developer Data Modeler tool for diagramming.

## Database

MariaDB 10.1.30

MariaDB is platform independent and will run on Windows, Mac, or Linux.   
HeidiSQL was used to interface with the database (HeidiSQL Version: 9.4.0.5125).

## SQL Editor

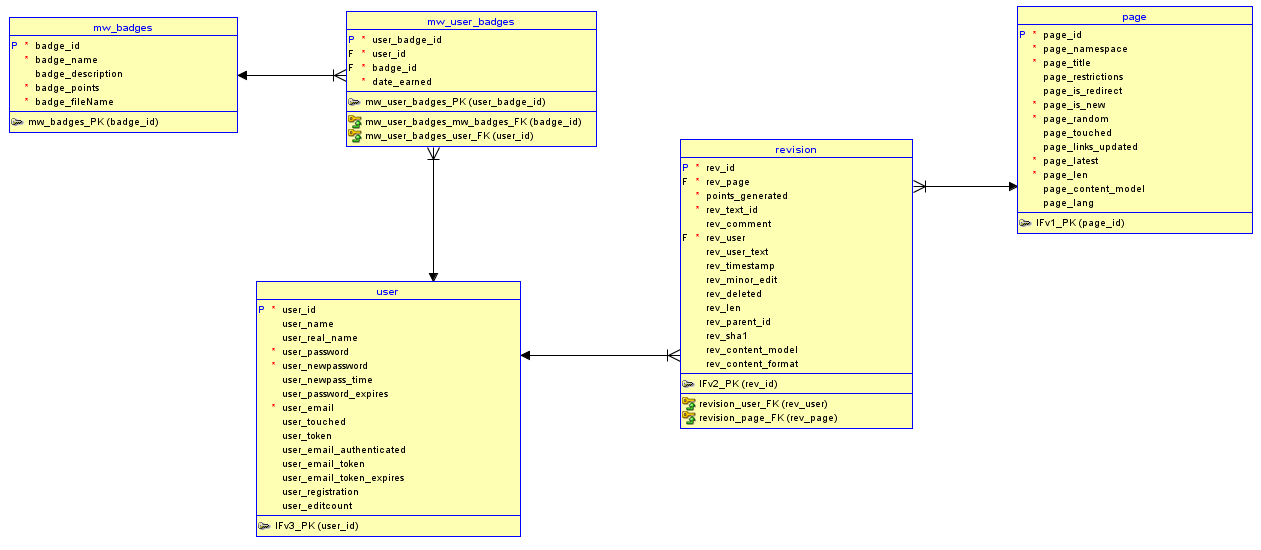
NotePad++.

## DDL and DML

SQL will be used for the DDL and DML. The information captured in the ER diagram will be used to generate SQL scripts to create the database tables. In addition, SQL will also be used to create DML scripts to insert data to be used for testing.

## ERD

The following ERD diagram contains two custom tables: **mw\_badges** and **mw\_user\_badges** and three tables that are part of the default MediaWiki database.



From the default MediaWiki database we will be using the **user, revision, and page** tables because users will be earning points and badges according to their contributions (edits/revisions to pages).

# Relationships

1. **mw\_badges** and **user**  
   There is a **M:M relationship** between **user** table and **mw\_badges** table because a user may have many badges and a badge may be assigned to many users. To overcome a many-to-many relationship situation we have created **mw\_user\_badges** table which will hold the user\_id as a Foreign Key from the user table, and badge\_id from the mw\_badges table.
2. **user** and **page**  
   There is a **M:M relationship** between **user** table and **page** table because a user may write on many pages and a page may be written by many users. To overcome a many-to-many relationship the application has created **revision** table which will hold the rev\_page as a Foreign Key from the page table, and rev\_user from the user table.

# Cardinalities/Business Rules

1. mw\_badges and user   
   A user can have zero, one, or many gamification badges, and a gamification badge may be assigned to zero, one, or many users.
2. user and page  
   A user my contribute to zero, one, or many pages, and page may be assigned to zero, one, or many contributors (users).

# Entities

All custom entities will start with the **mw** prefix, which refers to “media wiki”

1. Entity Name: **mw\_badges**

Entity Description: registered users in the MediaWiki database

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Type** | **Other Info** |
| badge\_id | to identify badges | INT | Primary Key  auto incremented |
| badge\_name | badge’s name | VARCHAR | Mandatory |
| badge\_description | badge’s description | VARCHAR |  |
| badge\_points | specifies the number of points needed to earn the badge | INT | Mandatory |
| badge\_fileName | badge's image name | VARCHAR | Mandatory |

1. Entity Name: **mw\_user\_badges**

Entity Description: registered users in the MediaWiki database

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Type** | **Other Info** |
| user\_badge\_id | to identify the use/badge/timestamp combination | INT | Primary Key  auto incremented |
| user\_id | to identify the user | INT | Foreign Key (user) Mandatory |
| badge\_id | to identify the user | INT | Foreign Key (wm\_badges) Mandatory |
| date\_earned | Date and time –including seconds when the badge was earned | BINARY | Mandatory |

1. Entity Name: **user (this table is generated by the MediaWiki application)**

Entity Description: registered users in the MediaWiki database

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Type** | **Other Info** |
| user\_id | to identify users | INT | Primary Key  auto incremented |
| user\_name | user’s screen name | VARCHAR | Unique  Mandatory |
| user\_real\_name | user’s real name | VARCHAR | Mandatory |
| user\_password | user’s password | TINYBLOB | Mandatory |
| user\_newpassword | user’s new password | TINYBLOB | Mandatory |
| user\_new\_pass\_time | datetime user’s new password was generated | BINARY |  |
| user\_email | User’s email address | TINYTEXT | Mandatory |
| user\_touched | last time user logged in | BINARY | Mandatory |
| user\_token | user’s token (if applicable) | BINARY | Mandatory |
| user\_email\_authenticated | datetime user’s email was authenticated | BINARY |  |
| user\_email\_token | token for user’s email (if applicable) | BINARY |  |
| user\_email\_token\_expires | datetime user’s email token expires (if applicable) | BINARY |  |
| user\_registration | datetime user registered on site | BINARY |  |
| user\_editcount | number of times user has edited a page | INT |  |
| user\_password\_expires | expiration date of user’s password (if applicable) | VARBINARY |  |

1. Entity Name: **revision (this table is generated by the MediaWiki application)**

Entity Description: edits and revisions users make to the page

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Type** | **Other Info** |
| rev\_id | identifies the revision | INT | Primary Key  auto incremented |
| rev\_page | identifies the page | INT | Foreign Key (user) Mandatory |
| points\_generated | helps track the number of points generated by the revision | INT | Mandatory |
| rev\_text\_id | assigns an id for the generated text | INT | Mandatory |
| rev\_comment | high level comment of what the page revision was about | VARBINARY |  |
| rev\_user | identifies the user by the ID | INT | Mandatory |
| rev\_user\_text | identifies the user by the name | VARCHAR | Mandatory |
| rev\_timestamp | time when the revision took place; it specifies year, month, day, hour, and minutes | BINARY | Mandatory |
| rev\_minor\_edit | identifies if users have marked the revision as “minor” | TINYINT | Mandatory |
| rev\_deleted | identifies if the revision has been deleted | TINYINT | Mandatory |
| rev\_len | counts the total number of characters –including spaces, special characters, etc. that are in the page | INT |  |
| rev\_parent\_id | Identifies the previous revision to the current one. | INT |  |
| rev\_sha1 | information set up by the database | VARBINARY | Mandatory |
| rev\_content\_model | information set up by the database | VARBINARY |  |
| rev\_content\_format | information set up by the database | VARBINARY |  |

1. Entity Name: **page (this table is generated by the MediaWiki application)**

Entity Description: stores page information

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Type** | **Other Info** |
| page\_id | identifies the page | INT | Primary Key  auto incremented |
| page\_namespace | this field has internal meaning for the application | INT | Mandatory  unique |
| page\_title | stores page’s title | VARCHAR | Mandatory unique |
| page\_restrictions | stores page’s restrictions | TINIBLOB | Mandatory |
| page\_is\_redirect | identifies if the page redirects to another page | TINYINT | Mandatory |
| page\_is\_new | identifies if it is a new page | TINYINT | Mandatory |
| page\_random | stores random number assigned to the page | DOUBLE | Mandatory |
| page\_touched | identifies the date (year, month, day, hour, minute) when the page was modified | BINARY | Mandatory |
| page\_links\_updated | stores the date when links were updated | VARBINARY |  |
| page\_latest | identifies the latest revision ID | INT | Mandatory |
| page\_len | counts the total number of characters in the page | INT | Mandatory |
| page\_content\_model | this field has internal meaning for the application | VARBINARY |  |
| page\_lang | stores the page’s language | VARBINARY |  |

# Assumptions and Special Considerations

1. The date\_earned field will be populated when the badge is actually earned and cannot be null.
2. Players may have multiple badges of the same type until they reach a higher level which they will be exchange for the higher level badge.

# 

# DDL Statements

Only the DDL statements that are not related to the default application’s database are going to be enumerated below.

Set Global Variables

/\* = = = = = = = = = = = =

= SET GLOBAL VARIABLES

= = = = = = = = = = = = \*/

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time\_zone = "+00:00";

Drop Statements

(If tables mw\_badges and mw\_user\_badges already exist)

/\* = = = = = = = = = =

= = DROP TABLES

= = = = = = = = = = = \*/

DROP TABLE IF EXISTS mw\_badges;

DROP TABLE IF EXISTS mw\_user\_badges;

Create Table Statements & Primary Key Constraints

(If tables don’t already exist.)

/\* = = = = = = = = = = = =

= = CREATE TABLE mw\_badges

= = = = = = = = = = = = = \*/

CREATE TABLE IF NOT EXISTS mw\_badges (

badge\_id int(2) unsigned NOT NULL AUTO\_INCREMENT,

badge\_name varchar(50) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL,

badge\_description varchar(500) CHARACTER SET utf8 COLLATE utf8\_bin,

badge\_points int(5) unsigned NOT NULL,

badge\_fileName varchar(50) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL,

PRIMARY KEY (badge\_id)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

/\* = = = = = = = = = = = =

= = CREATE TABLE mw\_user\_badges

= = = = = = = = = = = = = \*/

CREATE TABLE IF NOT EXISTS mw\_user\_badges (

user\_badge\_id int(15) unsigned NOT NULL AUTO\_INCREMENT,

user\_id int(15) unsigned NOT NULL,

badge\_id int(2) unsigned NOT NULL,

date\_earned binary(14) NOT NULL DEFAULT '\0\0\0\0\0\0\0\0\0\0\0\0\0\0',

PRIMARY KEY (user\_badge\_id)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

Create Foreign Key Constraints

/\* = = = = = = = = = = = =

= = CREATE Foreign Key Constraints for TABLE mw\_badges

= = = = = = = = = = = = = \*/

ALTER TABLE mw\_user\_badges

ADD CONSTRAINT badge\_id\_fk FOREIGN KEY (badge\_id)

REFERENCES mw\_badges(badge\_id)

ON DELETE CASCADE

ON UPDATE CASCADE;

ALTER TABLE mw\_user\_badges

ADD CONSTRAINT user\_id\_fk FOREIGN KEY (user\_id)

REFERENCES user(user\_id)

ON DELETE CASCADE

ON UPDATE CASCADE;

Add Column to the Revision Table

/\* = = = = = = = = = = = =

= = ADD COLUMN TO revision TABLE

= = = = = = = = = = = = = \*/

ALTER TABLE revision

ADD earned\_points INT(4) unsigned NOT NULL

AFTER rev\_page;