Thursday, March 16, 2017

Academic Paper Proofreading Website

Deliverable 2 – Database Implementation (Week 8)

Group 4

John Juele	15167798
Sophia Colgan	15159973
Eoghan Casey	15160513
Donagh Kelleher	15162788

Table of Contents

Introduction

In this document, I will be discussing the implementation of our database schema in our web development project. Within this report, I will be including:

- **Database Tables**
  - Each table will consist of:
    - Attribute list + datatypes
    - Purpose in web system
    - SQL statement for table

Github Link

Click here to access SQL file in our repository

Just in case it fails:

https://github.com/Coding-Chicken-Lover/Web-Development-Project/blob/master/Documents/Database%20Schemas/tables.sql

NOTE: No sample data will be added to any database tables with this SQL code.

Database Tables

`Users` table

TABLE ATTRIBUTES	DATATYPE
User_ID	INT
FirstName	VARCHAR(128)
LastName	VARCHAR(128)
Email	VARCHAR(128)
Subject	VARCHAR(126)
Rep_Points	INT
Password	VARCHAR(255)

The `Users` table is used to store information on each user of the website. Each user is identified and represented by their University ID which in this case is the `User_ID`

```
CREATE TABLE IF NOT EXISTS `Users` (
    `User_ID` int unsigned NOT NULL,
    `FirstName` varchar(128) NOT NULL,
    `LastName` varchar(128) NOT NULL,
    `Email` varchar(128) NOT NULL,
    `Subject` varchar(126) NOT NULL,
    `Rep_Points` int unsigned DEFAULT '0',
    `Password` varchar(255) NOT NULL,
    PRIMARY KEY (`User_ID`),
    UNIQUE KEY (`Email`)
);
```

`Tasks` table

TABLE ATTRIBUTES	DATATYPE
Task_ID	INT
Owner	INT
Date_Created	DATETIME
Title	VARCHAR(255)
Type	VARCHAR(20)
Description	VARCHAR(5000)
Pages	INT
Words	INT
Format	VARCHAR(10)

The `Tasks` table is used to store basic information on each task a user creates. Each task is identified and represented by their `Task_ID` and the ID of its `Owner`. Each User can publish multiple tasks at a time.

```
CREATE TABLE IF NOT EXISTS `Tasks` (
    `Task_ID` int unsigned NOT NULL AUTO_INCREMENT,
    `Owner` int unsigned NOT NULL,
    `Date_Created` datetime NOT NULL,
    `Title` varchar(255) DEFAULT NULL,
    `Type` varchar(20) DEFAULT NULL,
    `Description` varchar(5000) DEFAULT NULL,
    `Pages` int unsigned DEFAULT 0,
    `Words` int unsigned DEFAULT 0,
    `Format` varchar(10) DEFAULT NULL,
    PRIMARY KEY (`Task_ID`, `Owner`),
    FOREIGN KEY (`Owner`) REFERENCES `Users`(`User_ID`) ON DELETE CASCADE ON UPDATE
CASCADE
);
```

`Tags` table

TABLE ATTRIBUTES	DATATYPE
Tag_ID	INT
Title	VARCHAR(20)

The `Tags` table is used to store tags that a user can use to classify a task they have created. Each tag with a particular title is represented by their `Tag_ID`

```
CREATE TABLE IF NOT EXISTS `Tags` (
	`Tag_ID` int unsigned NOT NULL AUTO_INCREMENT,
	`Title` varchar(20) NOT NULL,
	PRIMARY KEY (`Tag_ID`)
);
```

`Task_Tags` table

TABLE ATTRIBUTES	DATATYPE
Task_ID	INT
Tag_ID	INT

The `Task_Tags` acts as a dependent entity between the `Tasks` table and the `Tags` table. It keeps track on what tag is connected to a particular task. Each task can only have 4 tags at a time.

```
CREATE TABLE IF NOT EXISTS `Task_Tags` (
	`Task_ID` int unsigned NOT NULL,
	`Tag_ID` int unsigned NOT NULL,
	FOREIGN KEY (`Task_ID`) REFERENCES `Tasks`(`Task_ID`) ON DELETE CASCADE ON
UPDATE CASCADE,
	FOREIGN KEY (`Tag_ID`) REFERENCES `Tags`(`Tag_ID`) ON DELETE CASCADE ON UPDATE
CASCADE
);
```

`Task_Status` table

TABLE ATTRIBUTES	DATATYPE
Task_ID	INT
Status	VARCHAR(20)
Claimant	INT
Rating	VARCHAR(10)

The `Task_Status` table stores the state of a task in a certain period of time. A task's possible states are:

- PENDING_CLAIM
- CLAIMED
- UNCLAIMED
- CANCELLED
- COMPLETE
- FAILED

```
CREATE TABLE IF NOT EXISTS `Task_Status` (
	`Task_ID` int unsigned NOT NULL,
	`Status` varchar(20) NOT NULL DEFAULT 'PENDING_CLAIM',
	`Claimant` int unsigned NOT NULL,
	`Rating` varchar(10) DEFAULT NULL,
	PRIMARY KEY (`Task_ID`),
	FOREIGN KEY (`Task_ID`) REFERENCES `Tasks`(`Task_ID`) ON DELETE CASCADE ON
UPDATE CASCADE,
	FOREIGN KEY (`Claimant`) REFERENCES `Users`(`User_ID`)
);
```

`Deadlines` table

TABLE ATTRIBUTES	DATATYPE
Task_ID	INT
Claim_D	DATETIME
Sub_D	DATETIME

The `Task_Status` table stores deadlines for a task. The `Claim_D` attribute stores the expiry date of a task in the task stream. The `Sub_D` attribute stores the date at which the claimant must submit his/her's work back to the task owner.

```
CREATE TABLE IF NOT EXISTS `Deadlines` (
	`Task_ID` int unsigned NOT NULL,
	`Claim_D` datetime NOT NULL,
	`Sub_D` datetime NOT NULL,
	PRIMARY KEY (`Task_ID`),
	FOREIGN KEY (`Task_ID`) REFERENCES `Tasks`(`Task_ID`) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

`Banned_Users` table

TABLE ATTRIBUTES	DATATYPE
Banned_User	INT
Banner	DATETIME
Date	DATETIME

The `Banned_Users` table keeps in account the users who have been banned by a moderator. It also stores the date of when the user got banned.

```
CREATE TABLE IF NOT EXISTS `Banned_Users` (
	`Banned_User` int unsigned NOT NULL,
	`Banner` int unsigned NOT NULL,
	`Date_Banned` datetime NOT NULL,
	PRIMARY KEY(`Banned_User`),
	FOREIGN KEY (`Banned_User`) REFERENCES `Users`(`User_ID`),
	FOREIGN KEY (`Banner`) REFERENCES `Users`(`User_ID`)
);
```

`Flagged_Tasks` table

TABLE ATTRIBUTES	DATATYPE
Task_ID	INT
Flagger	INT
Description	VARCHAR(15)
Review_Status	VARCHAR(10)
Date_Flagged	DATETIME

The `Flagged_Task` table stores the tasks that have been flagged. The table also includes the user that flagged the user, the reason of the flag, a flag indicating whether the task has been checked by a moderator and the date when the task got flagged.

```
CREATE TABLE IF NOT EXISTS `Flagged_Tasks` (
	`Task_ID` int unsigned NOT NULL,
	`Flagger` int unsigned NOT NULL,
	`Description` varchar(15) NOT NULL,
	`Review_Status` varchar(10) DEFAULT 'UNCHECKED',
	`Date_Flagged` datetime NOT NULL,
	PRIMARY KEY(`Task_ID`, `Flagger`),
	FOREIGN KEY(`Task_ID`) REFERENCES `Tasks`(`Task_ID`)
);
```