The Crocodile Database Manual

# Overview

This Crocodile website’s data is a combination of SQL database and an online database called Firebase. We will begin with a rundown of our SQL database which holds the items table. This table will hold all relevant data on the items added into the Crocodile facility for display. In SSMS, SQL Server Management System, a database admin will be able to manipulate the items table by adding, editing, or deleting any data that they want. Other suggested usages for this program to add more layers would be creating new tables such as a customer info table, a related items table, or manipulating existing tables to an admin’s preference. Below will be a rundown of our existing code and what it means.

**ER Diagram**

Table

Description automatically generated

Above is the ER Diagram for our Crocodile SQL Server database. It consists of 19 Rows of data and below is the relational schema that contains the metadata for the database.

Graphical user interface, table, Excel

Description automatically generated

The following table provides column names and descriptions of what their data will hold:

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Description | Column Name | Description |
| Item Key with solid fill | This is an autogenerated integer in sequential order that is assigned to each item. This is the Surrogate Key. | Accesion\_date | Provides the date of the item. Category can take nulls so it is optional. |
| Item\_ID | This is a user assigned integer that will provide simple look up for items in the database and can provide future linking. | Collector | Provides who is the item’s collector/owner. |
| QR\_code | This category provides the QR code created for the related item. | Item\_narrative | Provides the data for capturing item description and story. |
| Item\_name | Provides the name for the item. | Provenance | Provides earliest known history or place of origin. Category can take nulls so it is optional. |
| Item\_type | Provides the type of item. | Cross\_References | Provides any references to other items which can be put in note form. Category can take nulls so it is optional. |
| Maker | Provides who made the item. | Image\_url1,2,3,4 | Provides a url from our Firebase online storage for an image. |
| Size | Provides the size of the item if available. Category can take nulls so it is optional. | Video\_url | Provides a url from our Firebase online storage for a video. Category can take nulls so it is optional. |
| Condition | Provides the condition of the item. Category can take nulls so it is optional. |  |  |

## Creating the database

**Graphical user interface, text, application, email

Description automatically generated**

Lines 1-24 of this code consist of the creator tags and date in note form from line 1-3.

Lines 5-23 consist of code that selects the server-level permissions in order to create, edit or delete tables. It then finds the existing database and will drop the existing database so it will allow you to freely alter, add or delete any data without errors.

Lines 19-23 Creates the CrocodileDB database and selects that same database to use.

**Clearing data for manipulation**

Graphical user interface, text, application

Description automatically generated

Lines 25-37 consist of code to prepare for data manipulation by completely clearing the data from the Item table. This is written in a form of a stored procedure.

**Creating the Item table**

Table

Description automatically generated

Lines 39-68 in the form of a stored procedure, the following creates the tables and allows the admin to create column names and data types. It also allows you to choose if Nulls are allowed for the data. You can also rearrange column names as the admin sees fit.

\*This is the area where you can add more categories for the items for more detail.

In order to do this under create table in lines 46-62 you would add another line under or in between other column names in this form:

CREATE TABLE table\_name (

column\_name1 data\_type(size) constraint\_name,

, [new column name] (choose data type) (for data that can’t accept Nulls, write NOT NULL)

## Entering data into table

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Lines 72-98 consist of code to enter data into the table. Using a stored procedure, we begin by using the INSERT INTO function. The structure of this code allows the admin to pick which column name to insert data into by selecting the Item table and adding values. After coding the VALUES function an admin will insert data by using the following structure:

INSERT INTO CrocodileDB.dbo.Item VALUES (‘Item\_id’, ‘QR\_code’, ‘Item\_name’, ‘Item\_type’, ‘Maker’, ‘Size’, ‘Condition’, ‘Accesion\_date’, ‘Collector’, ‘Item\_narrative’, ‘Provenance’, ‘Cross\_references’, ‘Image\_url1’, ‘Image\_url2’, ‘Image\_url3’, ‘Image\_url4’, ‘Video\_url’)

Inside the ‘ ’ is where the admin will add data, the above figure shows which column each is identified with. If the admin chooses to edit or add columns, they will have to make sure to use the new or edited column names.

Important to notice that the ItemKey is autogenerated and it should not be added to the code.

## Executing stored procedures

Background pattern

Description automatically generated with low confidence

\*Please disregard line numbers for the above figure

Lines 100-103 is code to execute the three store procedures in the above lines. We have three store procedures, one to wipe old data if it exists, one to create the Item table and one to insert data into the table.

The SQL database has now been created and is ready to deploy for online services.

## Firebase

Firebase is the online storage that houses all the image and video files. It is linked to our SQL tables by the Firebase url associated for that media file and has been inserted into image url and video url in the Item table. To access the Firebase, account the information is as follows:

Access Site: <https://console.firebase.google.com/>

E-Mail: [CrocManagerMain@gmail.com](mailto:CrocManagerMain@gmail.com)

Password: CrocProject1775

Graphical user interface, application, Teams

Description automatically generated

To view, edit, add, or delete images or videos navigate to the console and on the left-hand side of the navigation window under Build, select Storage. Here you can upload your files and view or edit any existing ones. Please note that if you add images you will have to add the URL to the SQL database as explained under the SQL manual in section ‘Entering data into the table’.

**Captured E-mails**

Graphical user interface, application

Description automatically generated

Also within Firebase is the Crocodile website’s log-in data. To view all log-in information in the console navigate to Authentication which is on the left-hand side of the window. Under Users it will show all emails captured through the web application log-in feature of the website as well as creation date, last log-in date and a unique user identifier code.

Graphical user interface, application, Teams

Description automatically generated

Under Sign-in method it allows the admin to add new providers and also authorized domains to capture data from.

Graphical user interface, application, Teams

Description automatically generated

Under Templates it will allow the admin to control confirmation emails, password reset emails, and email address change emails which are sent to users after one of those actions are executed.

Firebase has many other features that can be implemented to the website, one of the other suggested usages is looking at the analytics and usage data of the site to see how much data is being pulled.