Bright Light Systems 1.0

Database Design

Document version 1.0

INTRODUCTION:

Before reading this document, it is **strongly recommended** to familiarize with the Bright Light Systems Requirements and the System Design.

This document is a part of the Bright Light Systems Design Documentation. It provides information about the Bright Light Systems 1.0 database structure, description of the tables and their columns and entries.

The Bright Light Systems' database is a relational database that **locally** stores information about users, bridges, light bulbs, bulb's settings, and all of the application's other data. At this moment the database includes **10** tables.

The database was designed such that the process of adding new components would be easy, and that the final result should not affect the initial model.

Since Bright Light Systems interfaces with the database by itself, an end user should not have to worry about its structure. However, he/she through the provided interface will be able to specify how and where to back up the application's data, as well as restore it if the data was lost.

The type of database and SQL Scripting incorporated by the Bright Light Systems' build will depend on the device being used. Table 1 shows databases and devices that will be supported in Bright Light Systems 1.0.

Database supported	End device
SQLite	• Android phones • Android tablets
MySQL	• Apple iPhone • Apple iPad

Table 1. Supported databases and devices

This document uses MySQL data types to describe table columns. Please note that column types may vary depending on the database. While reading, please also refer to the documents below if needed.

- **Bright Light Systems Requirements v. 1.0**
- **Bright Light Systems System Design v. 1.0**

CORE FUNCTIONALITY GOALS:

The Bright Light Systems 1.0 database will provide the ability for users to manage their bridges and connect them to light bulbs. The user will be able to organize them by groups as well as create themes and traits. These can be applied to an individual bulb or a set of light bulbs, immediately or later on. The user can also setup conditions that would be used to automatically apply specified themes and/or states on certain light bulbs' groups.

Actions on tables:

- Create – creates an entry in the database.
- Read - reads the data from table(s); optionally based on specified conditions.
- Update – updates existing entry; optionally based on specified conditions
- Delete – deletes an entry from the table; optionally based on specified conditions.

Database Maintenance:

Because the Bright Light Systems' database stores data locally, it is important to support a mechanism that will back up and restore all information when needed. This will be done by creating logical backups, which will involve reading the data and writing it to .CSV files. The files will be emailed to the user using his or her email address.

DATABASE DIAGRAM:

Diagram 1 provides a visual overview of the Bright Light Systems' database model and the relations between the tables.

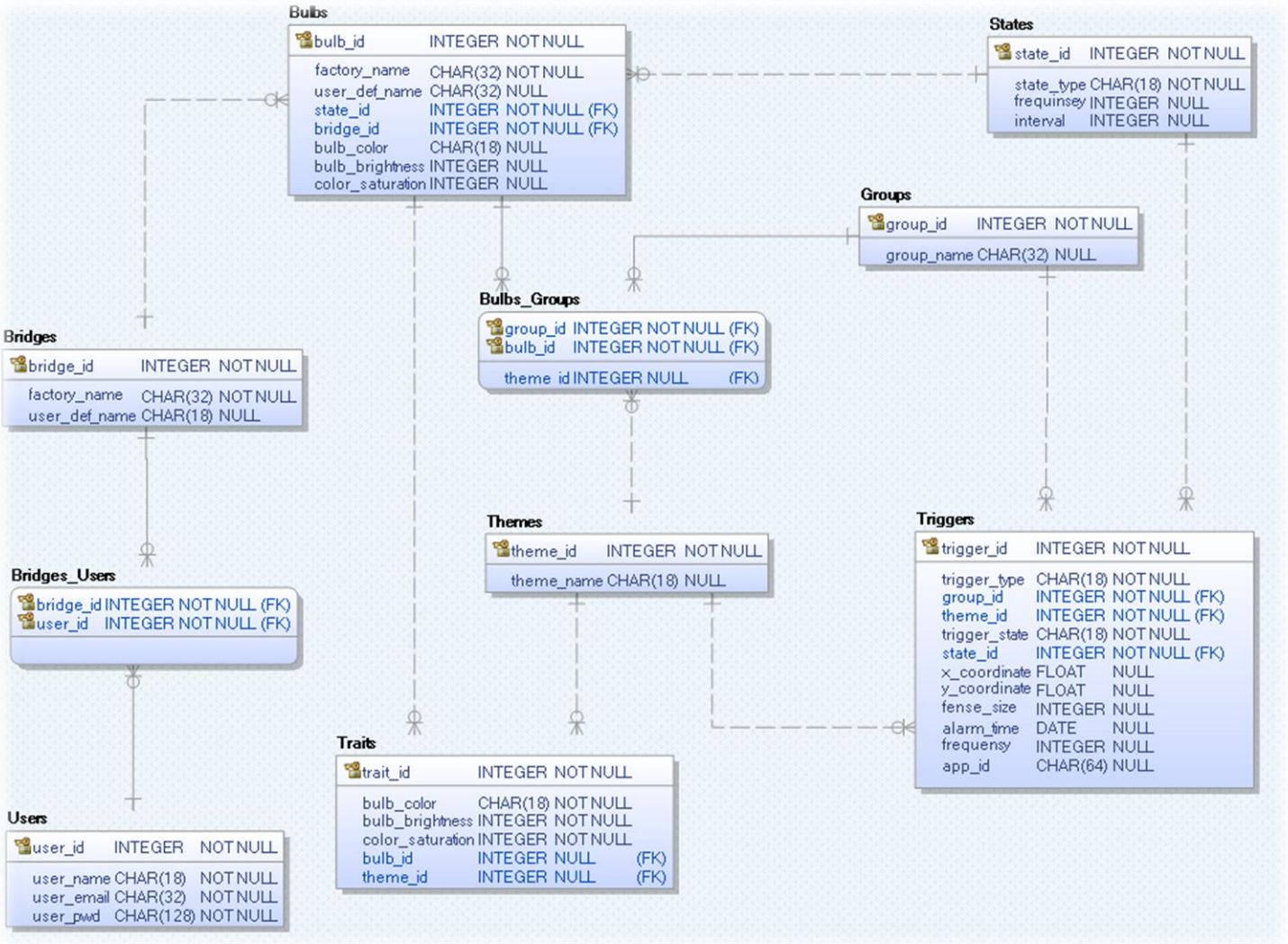

Diagram 1. Bright Lights Systems.1.0 database model

TABLES OVERVIEW:

This section provides an overview of the tables, their columns and entries. Please review used abbreviations for some key words below:

PK – primary key

FK – foreign key

AUTO - auto incremented value, where **s**:**n i:c** means first entry of the column starts with value **n** incremented by value **c** for the next entry. Example: s:2 i:3 (start with 2 increment by 3)

Users

Contains basic information about users. All fields are required.

Name	Type	Null	Key	Default	Additional
user_id	INTEGER		PK/AUTO	s:1 i:1	
user_name	CHAR(18)				
user_email	CHAR(18)				
user_pwd	CHAR(128)				Password. Hashed before adding to the table.

Bridges

Describes bridges. The table forms a many-to-many relationship with the ***Users*** table. Such design allows a user to control many bridges, while each bridge can be controlled by many users. The relationship is resolved in the ***Bridges_Users*** table. The ***Bulbs*** table refers to the ***Bridges*** table as well. Thus, if an entry in this table is deleted, the corresponding information about the connected bridge from the ***Bulbs*** table will be deleted as well.

Name	Type	Null	Key	Default	Additional
bridge_id	INTEGER		PK/AUTO	s:1 i:1	
factory_name	CHAR(32)				Serial number requested with Hue API
user_email	CHAR(18)	YES			

Bridges_Users

The resolution for the many-to-many relationship between ***Users*** and ***Bridges***.

Name	Type	Null	Key	Additional
bridge_id	INTEGER		FK	Combination of these two values forms the PK.
user_id	INTEGER		FK	

Bulbs

Contains information about light bulbs **assigned** to a specific bridge. Each entry describes an individual light bulb which can be assigned only to one bridge. If a bridge has been deleted, all light bulbs assigned to that bridge have to be deleted as well. Each light bulb has its own state, which is described in the ***States*** table and referenced by the **state_id** column. The state of the light bulb may be **on** or **off,** among other values, that are described in the ***States*** table.

Name	Type	Null	Key	Default	Additional
bulb_id	INTEGER		PK/AUTO	s:1 i:1	
factory_name	CHAR(32)				
user-def-name	CHAR(32)	YES			
state_id	INTEGER		FK	ON	Default value of ON state
bridge_id	INTEGER		FK		
bulb_color	CHAR(18)	YES			These values contain current settings of a light bulb. If the light bulb was off, these settings may be used to restore the bulb's state after turning it back on. If any of the values are Null, the default settings for that value requested through the Hue API have to be used.
bulb_brightness	INTEGER	YES			
color_saturation	INTEGER	YES			

States

Defines states that a light bulb can take. This version of the database will support 4 states: ON/OFF/BLINK/UNREACHABLE. User can modify **only** the number of blinks and the length of an interval between each blink in this table.

Name	Type	Null	Key	Default	Additional
state_id	INTEGER		PK/AUTO	s:1 i:1	
state-type	CHAR(18)			ON/OFF/BLINK/UNREACHABLE	One of these 4
frequency	INTEGER	YES			NULL values to record ON/OFF/UNREACHABLE states.
interval	INTEGER	YES			

Groups

Contains information about a group. ***Bulbs*** and ***Groups*** *are* designed to have a many-to-many relationship, since each bulb can be a part of many groups and each group can have many bulbs in it. This relationship is resolved in ***Bulbs_Groups*** table.

Name	Type	Null	Key	Default
bridge_id	INTEGER		PK/AUTO	s:1 i:1
group _name	CHAR(32)	YES		

Bulbs_Groups

The resolution for the many-to-many relationship between ***Groups*** and ***Bulbs***.

Name	Type	Null	Key	Additional
group_id	INTEGER		FK	Combination of these two values forms the PK.
bulb_id	INTEGER		FK	
theme_id	INTEGER	YES	FK	This value may reference to a theme that is assigned to the group.

Themes

Contains information about themes. This table is referenced by the ***Traits*** table.

Name	Type	Null	Key	Default
theme_id	INTEGER		PK/AUTO	s:1 i:1
theme _name	CHAR(18)	YES		

Traits

Describes light bulbs' traits which include color, brighness, and density. Each trait may be assigned to a particular light bulb and/or a theme.

Name	Type	Null	Key	Default
trait_id	INTEGER		PK/AUTO	s:1 i:1
bulb_color	CHAR(18)			
bulb_brightness	INTEGER			
color_ saturation	INTEGER			
bulb_id	INTEGER	YES	FK	
theme_id	INTEGER	YES	FK	

Triggers

Contains information about user-defined triggers. This version of the database supports storing data for the following triggers:

- time-based triggers such as alarms and timers
- location based triggers
- notification based triggers

Name	Type	Null	Key	Default	Additional
trigger_id	INTEGER		PK/AUTO	s:1 i:1	
trigger_type	CHAR(18)			TIMER/LOCATION/NOTIFICATION	One out of these 3
group_id	INTEGER		FK		Group of light bulbs that will be involved in the event
theme_id	INTEGER		FK		Theme that will be applied on the group of light bulbs
trigger_state	CHAR(18)			ENABLED/DISABLED	One out of these 2
state_id	INTEGER		FK		State that all light bulbs have to take when event is triggered
x_coordinate	FLOAT	YES			Coordinates for the location trigger
y_coordinate	FLOAT	YES			
fense_size	INTEGER	YES			
alarm_time	DATE	YES			Values for the time based trigger
frequency	INTEGER	YES			
app_id	CHAR(64)	YES			Value for the notification trigger

CONCERNS:

This section provides with the list of possible issues that have to be considered during implementation.

- **Synchronization**
  The current physical state of a light bulb and the current logical state of the same light bulb, recorded in the database, may be out of synch. This could be caused by several physical factors such as light switch off, power loss, or light bulb breakage. In order to keep the data in synch, it is important to update the following ***Bridges*** and ***Bulbs*** tables every time the user launches the application.

- **Data Conservation**
  In addition to what is described in the Core Functionality Goals database maintenance section, it is important to prevent data loss or corruption during updates. It is crucial that the user's data stored in the older version of the database will safely transfer to the new version.

FUTURE CHANGES:

This section describes the Bright Light Systems database components that might be changed or added to support implementation of new features. Please note that following is a brief overview. The full specification of the new features will be documented in the future versions of the Requirement Document.

Table: States

This table might be expanded or completely redesigned to support additional states that a light bulb can take. As an example, one of the planned states to implement in the future is *Gradient.* The Gradient state will describe a transition between specified colors and the light bulb brightness over a specified time.

Table: Traits

At the moment created light bulb settings can be assigned to one particular bulb. In the feature it will be redesigned so that the user can assign the same trait to many different bulbs, and to support features such as “Recently Used” or “Favorite”.

Pattern Analyzer:

The Pattern Analyzer is a feature that analyzes how the user uses Bright Light Systems. Based on the history of choices, the analyzer will create a better user experience by implementing Smart Suggestions and Smart Events. For example if the user from the application dims the lights at certain hour, the analyzer after a period of time will start recognizing that and will dim the lights for him/her automatically. The database will need to be redesigned to support this functionality. More information about this will be available in the future versions of this document.