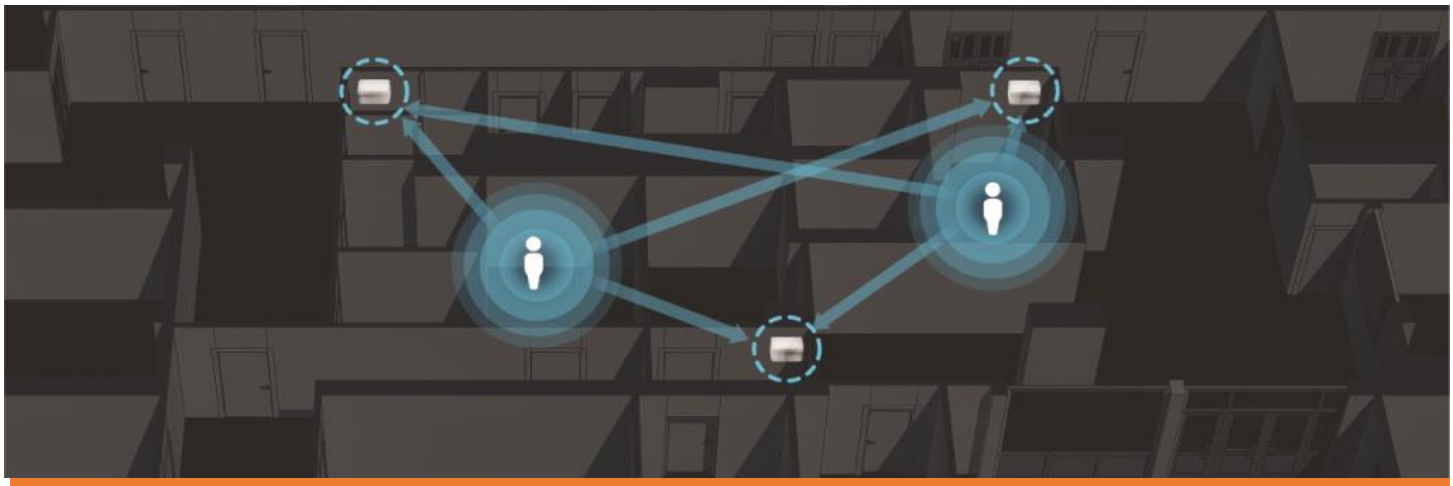
Powered by MySQL

ILLINOIS INSTITUTE OF TECHNOLOGY

Indoor Location Project

DATABASE DESIGN DOCUMENT (DDD)



Chakib LJAZOULI
cljazouli@hawk.iit.edu
06/29/2017

Table of Contents

1- Introduction:

The goal of this project is to provide a 911 operator with the indoor location of a person placing a 911 emergency call from a mobile phone. The indoor location must include the floor and room number where the caller is located. The motivation for this project is (1) a recent FCC report and order requiring PSAPs to "accurately identify the location of wireless 911 callers when the caller is indoors," as well as (2) the decision by a voluntary coalition of the major wireless carriers under the administration of the CTIA to create a demonstration of a system for providing such information.

The data are stored in MySQL 5.6.34 on a *db.t2.micro* instance in Amazon Web Services. It is Publicly Accessible. This design is to allow easy data entry access and querying to everyone working on the project or aiming to help us improve it around the world.

The driving philosophy behind the database design was to have an efficient, normalized database that would be easy to maintain and expand, as well as allow easy data entry and access.

1.1- Document Objectives

This DDD for the Indoor Location Project backend has the following objectives:

- To describe the design of the databases related to the project, that are, a collection of related data stored in many tables in a manner that can be accessed by users or computer programs via a database management system (DBMS). It also describes the software units used to access or manipulate the data.
- To serve as the basis for implementing the database and related software units. It provides the acquirer visibility into the design and provides information needed for software support.

1.2- Intended Audiences

This DDD is intended for the following audiences:

- Technical reviewers, who must evaluate the quality of this document.
- Indoor Location Project developers including:
  - Architects, whose overall architecture must meet the requirements specified in this document.
  - Designers, whose design must meet the requirements specified in this document.
  - Programmers, whose software must implement the requirements specified in this document.
  - Testers, whose test cases must validate the requirements specified in this document.

1.3- References

This DDD refers to the following references:

- Indoor Location Project Glossary
- Indoor Location Software Related Requirements Specifications
- Roadmap requirements
- NENA i3 standards

1.4- Database Overview

This database fills the following purposes:

- General nature of the database: MySQL database in an EC2 instance on AWS (Amazon web services.
- Business context of database: The system consists of an Array of Bluetooth Low Energy (BLE) beacons and a Database that together provide the indoor location, and a Smartphone Application that obtains the location information and makes an IP-based call, including the indoor location information, to a Next Generation 9-1-1 Public Safety Answering Point (PSAP.) The Array and the Database together constitute a platform that can be used by developers to create other applications that use its data.
- History of development: A primary version of the database was created to fits the need of the project. As the project and needs have evolved, the following database design was created from scratch to fill all the new requirements and to be prepared and easily modified for future changes in the project.
- Intended use: The database alteration will be used only for the Indoor Location Project purposes, and the data stored in it can be available for everyone willing to contribute to the project or for data analysis.
- Maintenance objectives: Achieving the best performance for the Indoor Location project system. Reducing the risks resulted from all the software related program's operations to minimum levels.
- Deployment location: EC2 instance on AWS accessible via the url: accesspointlocationserver.cbkzo1niupv4.us-east-1.rds.amazonaws.com in the port: *3306*

1.5- Document Overview

This DDD is organized into the following sections:

Introduction, which introduces the database design for the Indoor Location Project to its readers including referenced documents and an overview of the database including definition, business goals and context.

Database Overview, which provides a high-level description of the database including its definition, business goals, and context.

Database-wide design decisions, which provides decisions about the databases behavioral design (how it will behave, from a user's point of view, in meeting its requirements, ignoring internal implementation) and other decisions affecting further design of the database.

Detailed database design, which will contain a section for each design level (conceptual, internal, logical, physical).

2- Database-wide Design Decisions

This section documents decisions about the databases behavioral design (how it will behave, from a user's point of view, in meeting its requirements, ignoring internal implementation) and other decisions affecting further design of the database.

2.1- Interfaces

User can interact with the database using an Indoor Location server, by using multiple APIs built to fits all the database user's needs. Users can refer to the APIs documentation to see the request, inputs and outputs that the database is expecting.

2.2- Behavior

The design decision for the Indoor Location database was to separate types of tables into 3 different categories. Information related to beacons, information related to sergents, and a third category that is related to both.

A code color is used on the UML diagram to show users which category the table is in.

2.3- Appearance / Naming

There are coding standards on each programming language but in terms of MySQL I haven't found any universal coding practice that everyone follows so I considered different open source framework and popular NodeJS based software and found some generally applied standards on naming so I am trying to summaries it in this section of the DDD.

Properly design MySQL with proper naming conventions will help to write SQL query faster, helps to remove confusions and conflicts both on queries and programming language.

The naming convention rules used here are:

- Use lowercase: Will help on speed typing, avoid mistakes dues to case sensitivity etc.
- No space – use underscore instead

- No numbers in name only alpha English characters
- Valid understandable names
- Name should be self-explanatory
- Names should not be more than 64 characters.
- Name can be both singular and plural but database represent one database so it should be singular if possible.
- Lower case table name: MySQL is usually hosted in Linux server which is case sensitive so for best practice table name should be all lower case.
- Table name is Singular:
  We think table holds so many things like user table holds many users in the table, so name should be plural but table is a single entity as Model is only one so it's odd to have plural table name. So, name your table like user, invoice, comment.
- Prefix in table name: I have seen many times that table name has prefix usually database name or project name. Some time it is necessary to have prefix as in hosting environment we have many tables in one database to overcome limitation of database by hosting provider.
- Choose short and one or two words as possible.
- Field names should be understandable.
- Primary column name: Primary key can be id, or table name _id.
- Avoid using reserve word as field name: order, date, name are reserve word for database avoid using it. You can add prefix to these names to make it understandable like user_name, signup_date e.t.c
- Avoid using column with same name as table name. This can cause confusion while writing query.
- Do define foreign key on database schema.
- Foreign key column must have table name with their primary key, eg: blog_id represents foreign key id from table blog.
- Avoid semantically – meaningful primary key names. A classic design mistake is creating a table with primary key that has actual meaning like 'name' as primary key. In this case if someone change his name then relationship with another table will be effected and name can be repetitive (not unique).

2.4- DBMS Platform

MySQL is the world's most popular open source database, enabling the cost-effective delivery of reliable, high-performance and scalable Web-based and embedded database applications. It is an integrated transaction safe, ACID-compliant database with full commit, rollback, crash recovery, and row-level locking capabilities. MySQL delivers the ease of use, scalability, and high performance, as well as a full suite of database drivers and visual tools to help developers and DBAs build and manage their business-critical MySQL applications. MySQL is developed, distributed, and supported by Oracle, and the latest information about MySQL software can be found on the MySQL Web site. The MySQL database provides the following features:

- High Performance and Scalability to meet the demands of exponentially growing data loads and users.
- Self-healing Replication Clusters to improve scalability, performance and availability.
- Online Schema Change to meet changing business requirements.
- Performance Schema for monitoring user- and application-level performance and resource consumption.
- SQL and NoSQL Access for performing complex queries and simple, fast Key Value operations.
- Platform Independence giving you the flexibility to develop and deploy on multiple operating systems.
- Big Data Interoperability using MySQL as the operational data store for Hadoop and Cassandra.

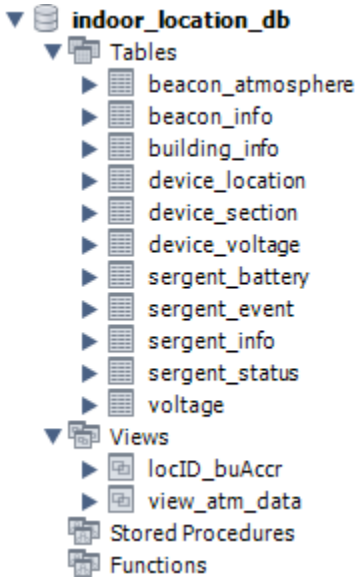
3- Detailed Database Design

This section describes the actual design of different databases at varying levels of abstraction. A subsection for each of conceptual, internal, logical and physical levels.

3.1- Indoor Location schema description and Diagram

3.1.1- Description

The following figure shows the Indoor Location Schema with all the tables and views in it:

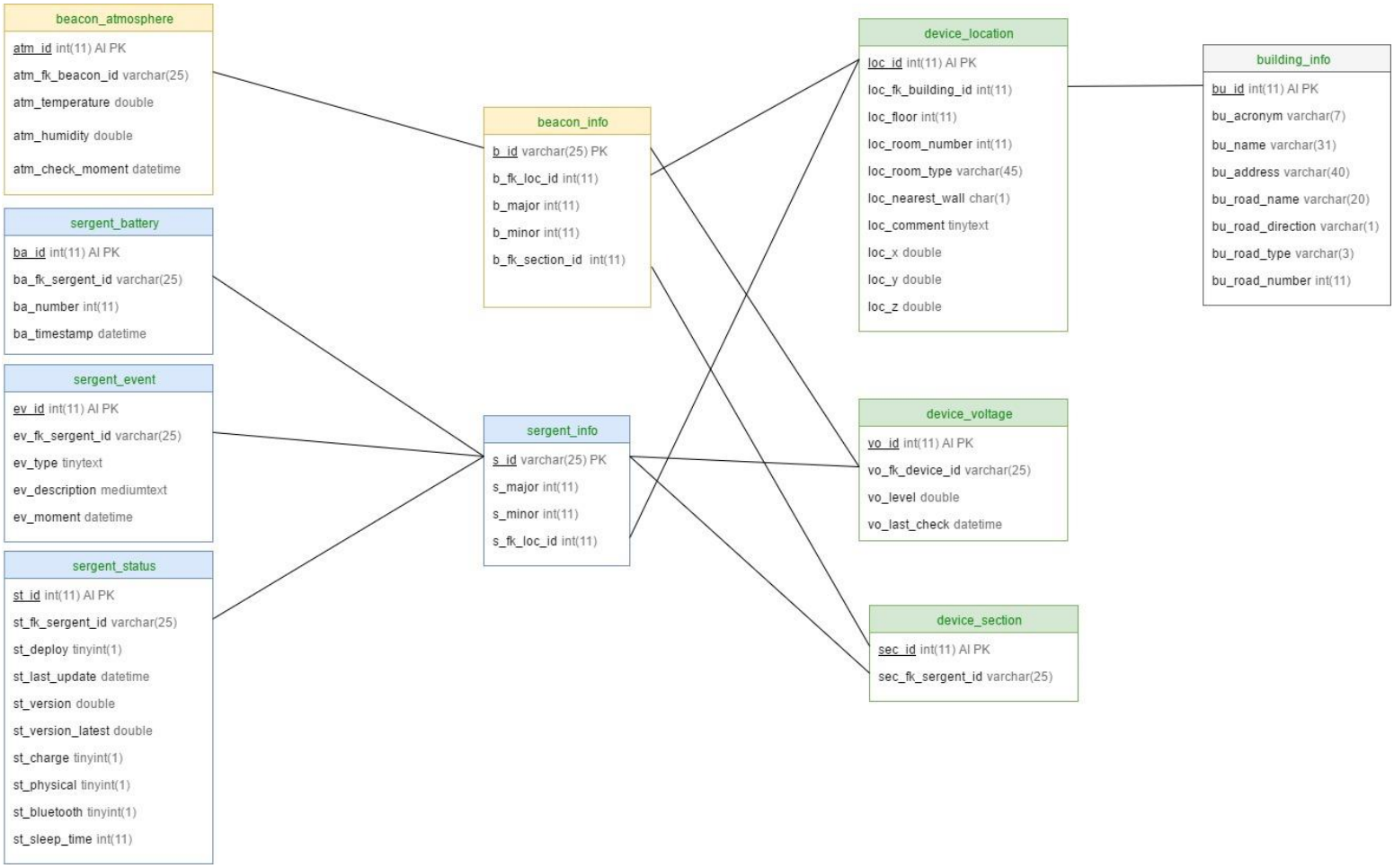


3.1.2- Diagram

The following figure shows the UML diagram of the Indoor Location database schema.

The diagram is available online, in the following GitHub repository:

https://github.com/IIT-RTC-Lab/Indoor_location_database



The tables in blue are related to the sergents information.

The tables in Yellow are related the beacons information.

The tables in green are related to both the sergents and the beacons.

3.2- Indoor Location Tables description:

In the following screenshots, the following abbreviations will be used:

PK - Primary Key

NN - Not Null

BIN - Binary (stores data as binary strings. There is no character set so sorting and comparison is based on the numeric values of the bytes in the values.)

UN - Unsigned (non-negative numbers only. so, if the range is -500 to 500, instead its 0 - 1000, the range is the same but it starts at 0)

UQ - Create/remove Unique Key

ZF - Zero-Filled (if the length is 5 like INT(5) then every field is filled with 0's to the 5th value. 12 = 00012, 400 = 00400, etc.)

AI - Auto Increment

G - Generated column. i.e. value generated by a formula based on the other columns

This section is the overview of all the tables created during the Indoor Location project. It is followed by specific information of what is in each table.

3.2.1- beacon_atmosphere

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
atm_id	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
atm_fk_beacon_id	VARCHAR(25)	☐	☑	☐	☐	☐	☐	☐	☐	
atm_temperature	DOUBLE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
atm_humidity	DOUBLE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
atm_check_moment	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP

This table defines the temperature and humidity of an Axa Beacon.

atm_id: The primary key

atm_fk_beacon_id: the foreign key to link with beacon ID

atm_temperature: temperature in F

atm_humidity: humidity in percentage

atm_check_moment: time stamp when the check was done.

3.2.2- Beacon_info

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
b_id	VARCHAR(25)	☑	☑	☐	☐	☐	☐	☐	☐	
b_major	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	'101'
b_minor	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
b_fk_loc_id	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
b_fk_section_id	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

This table is used to define all the information related to define and locate the beacon.

b_id: table primary key, which represents the MAC address of a beacon

b_major: Number defining the type of a beacon

b_minor: Number defining the beacon among the other beacons of the same type

b_fk_loc_id: foreign key to link with the id of the beacons location

b_fk_section_id: the section where the beacon is located (every section have a number of beacons in it)

3.2.3- building_info

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
bu_id	INT(11)	☑	☑	☑	☐	☐	☐	☑	☐	
bu_acronym	VARCHAR(7)	☐	☑	☐	☐	☐	☐	☐	☐	
bu_name	VARCHAR(31)	☐	☑	☐	☐	☐	☐	☐	☐	
bu_address	VARCHAR(40)	☐	☑	☐	☐	☐	☐	☐	☐	
bu_road_name	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
bu_road_direction	VARCHAR(1)	☐	☑	☐	☐	☐	☐	☐	☐	
bu_road_type	VARCHAR(3)	☐	☑	☐	☐	☐	☐	☐	☐	
bu_road_number	INT(11)	☐	☑	☐	☐	☐	☐	☐	☐	

This tables contains the information of a building.

bu_id: primary key of the building

bu_acronym: building acronym (example: SB for Stuart Building)

bu_name: Name of the building

bu_address: physical address of a building

bu_road_name: Road name

bu_road_direction: Direction (N, S, E or W)

bu_road_type: type of the road (Street, Avenue ...etc.)

bu_road_number: the number of the road

3.2.4- device_location

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
loc_id	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
loc_fk_building_id	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
loc_floor	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
loc_room_type	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
loc_room_number	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
loc_nearest_wall	CHAR(2)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
loc_comment	TINYTEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
loc_x	DOUBLE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
loc_y	DOUBLE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
loc_z	DOUBLE	☐	☐	☐	☐	☐	☐	☐	☐	NULL

This table is used to define the information concerning the location of a device. It can be the location of a beacon or the location of a sergent.

loc_id: the primary key of the device location

loc_fk_building_id: the foreign key to link with the building ID

loc_floor: the floor number

loc_room_type: the type of the room (example: bathroom)

loc_room_number: the number of the room if available

loc_nearest_wall: the nearest wall where the beacon is located

loc_comment: a comment if there is one about the location

loc_x: cartesian coordinate x

loc_y: cartesian coordinate y

loc_z: cartesian coordinate z

3.2.5- device_section

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
sec_id	INT(11)	☑	☑	☐	☐	☐	☐	☐	☐	
sec_fk_sergent_id	VARCHAR(25)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

This table is used to determine the section and the beacon that are defined in this section with the sergent responsible for it.

sec_id: Primary key of the section

sec_fk_sergent_id: foreign key to link with surgent responsible for the section

3.2.6- device_voltage

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
vo_id	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
vo_fk_device_id	VARCHAR(25)	☐	☑	☐	☐	☐	☐	☐	☐	
vo_level	DOUBLE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
vo_last_check	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL

This table is used to determine the voltage of a sergent or the voltage of a beacon to see if the device is charged or not.

vo_id: primary of the table

vo_fk_device_id: foreign key to link with the device primary key

vo_level: value in volt to determine if the device have enough battery

vo_last_check: the time stamp when the check was established

3.2.7- sergent_battery

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ba_id	INT(11)	☑	☑	☑	☐	☐	☐	☑	☐	
ba_fk_sergent_id	VARCHAR(25)	☐	☑	☐	☐	☐	☐	☐	☐	
ba_number	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
ba_timestamp	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL

This tables contains information related to the physical battery of a sergent.

ba_id: the primary key of the battery

ba_fk_sergent_id: the foreign key to link with the sergent

ba_number: the given number of a battery

ba_timestamp: the timestamp when the battery was put in a sergent

3.2.8- sergent_event

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ev_id	INT(11)	☑	☑	☑	☐	☐	☐	☑	☐	
ev_fk_sergent_id	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
ev_type	TINYTEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
ev_description	MEDIUMTEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
ev_moment	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP

This table contains information about all the event that happens on a sergent.

ev_id: the primary key of the evnt

ev_fk_sergent_id: the foreign key to link with the id of the sergent concerned

ev_type: type of the event that happened

ev_description: the description of the event happening

ev_moment: the time stamp when the even happened

3.2.9- sergent_info

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
s_id	VARCHAR(25)	☑	☑	☐	☐	☐	☐	☐	☐	
s_major	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
s_minor	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
s_fk_loc_id	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

This table contains all the information needed to determine a sergent.

s_id: the primary key of a sergent which is also his MAC address of the BTE module

s_major: a number that defines the type of the sergent

s_minor: number of the sergent among the sergents of the same type

s_fk_loc_id: the foreign key to link with its location id

3.2.10- sergent_status

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
st_id	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
st_fk_sergent_id	VARCHAR(25)	☐	☑	☐	☐	☐	☐	☐	☐	
st_deploy	TINYINT(1)	☐	☐	☐	☐	☐	☐	☐	☐	'0'
st_last_update	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL
st_version	DOUBLE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
st_version_latest	DOUBLE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
st_charge	TINYINT(1)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
st_physical	TINYINT(1)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
st_bluetooth	TINYINT(1)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
st_sleep_time	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	'86400'

This table contains all the information related to the status of a sergent.

st_id: the primary key of the sergent status

st_fk_sergent_id: the foreign key to link with the concerned sergent

st_deploy: to see if the sergent is deployed or not

st_last_update: time stamp of the last update

st_version: the current version of the sergent software

st_version_latest: the latest version of the sergent software

st_charge: to see if the sergent is charged or not

st_physical: to see if the sergent is broken or not

st_bluetooth: to see if the Bluetooth module of the sergent is on

st_sleep_time: te sleeptime in seconds of the sergent to save battery