**Highway Health: Database Design**

Database named HIGHWAYHEALTH contains table(s) GPS and WEATHER\_HISTORICAL.

**GPS table design:**

Columns:

* GPS\_ID : Integer that auto-increments and is the Primary Key.
* NAME : VARCHAR allows for 100 characters, Not Null. *Street name.*
* TYPE : VARCHAR allows for 5 characters, Not Null. *Street Type (US, SH, LS, TL).*
* LAT : VARCHAR allows for 20 characters, Not Null. *Latitude.*
* LON : VARCHAR allows for 20 characters, Not Null. *Longitude.*
* *Rows containing GPS\_ID, LAT, and LON should be Unique.*

**WEATHER\_HISTORICAL table design:**

Columns:

* W\_ID : Integer that auto-increments and is the Primary Key.
* REF\_ID : Integer that is the foreign key associated with GPS table Primary Key. Is also part of the Primary Key.
* DATETIME\_CST : VARCHAR allows for 20 characters, Not Null. *Date and time in CST.*
* LAT : VARCHAR allows for 10 characters, Not Null. *Latitude.*
* LON : VARCHAR allows for 10 characters, Not Null. *Longitude.*
* WEATHER\_ID : Integer. *Identifies a weather type defined by OpenWeatherMaps.*
* WEATHER\_DESC : VARCHAR allows for 100 characters, Not Null. *Description of the weather type identified in WEATHER\_ID.*
* WEATHER\_ICON : VARCHAR allows for 5 characters, Not Null. *Icon used for map.*
* TEMP\_F : Decimal with a full length of 5 digits and 2 decimal points. *Temp in Fahrenheit.*
* HUMIDITY\_PERCENT : Integer. *Humidity in percent.*
* VISIBILITY\_M : Integer. *Visibility in meters.*
* WINDSPEED\_MPH : Integer. *Windspeed in miles per hour.*
* DANGER\_LEVELS : VARCHAR allows for 20 characters, Not Null. *Level of danger on roads based on above data. ”IGNORE,” “MODERATE,” “DANGER”*

**Database Schema**

**Table

Description automatically generated**

**Create Database MYSQL Script**

Create database and use:

CREATE DATABASE IF NOT EXISTS HIGHWAYHEALTH;

USE HIGHWAYHEALTH;

Add tables to database:

CREATE TABLE IF NOT EXISTS GPS

(GPS\_ID INT NOT NULL AUTO\_INCREMENT,

NAME VARCHAR(100) NOT NULL,

TYPE VARCHAR(5) NOT NULL,

LAT VARCHAR(10) NOT NULL,

LON VARCHAR(10) NOT NULL,

PRIMARY KEY (GPS\_ID),

UNIQUE KEY u\_latlong (LAT,LON));

CREATE TABLE IF NOT EXISTS WEATHER\_HISTORICAL

(W\_ID INT NOT NULL AUTO\_INCREMENT,

REF\_ID INT NOT NULL,

DATETIME\_CST VARCHAR(20) NOT NULL,

LAT VARCHAR(10) NOT NULL,

LON VARCHAR(10) NOT NULL,

WEATHER\_ID INT NOT NULL,

WEATHER\_DESC VARCHAR(100) NOT NULL,

WEATHER\_ICON VARCHAR(5) NOT NULL,

TEMP\_F DECIMAL(5,2) NOT NULL,

HUMIDITY\_PERCENT INT NOT NULL,

VISIBILITY\_M INT NOT NULL,

WINDSPEED\_MPH INT NOT NULL,

DANGER\_LEVELS VARCHAR(20) NOT NULL,

PRIMARY KEY (W\_ID, REF\_ID),

FOREIGN KEY (REF\_ID) REFERENCES GPS (GPS\_ID))

~~Add triggers: (MAY NOT NEED)~~

~~DROP TRIGGER IF EXISTS prevent\_User\_UserID\_Update;~~

~~DELIMITER $$~~

~~CREATE TRIGGER prevent\_User\_UserID\_Update~~

~~BEFORE UPDATE ON USER~~

~~FOR EACH ROW~~

~~BEGIN~~

~~IF (NEW.USER\_ID <> OLD.USER\_ID)~~

~~THEN~~

~~SIGNAL SQLSTATE '45000'~~

~~SET MESSAGE\_TEXT = 'You cannot update a USER\_ID in the USER table.';~~

~~END IF;~~

~~END;~~

~~$$~~

~~DELIMITER ;~~

~~DROP TRIGGER IF EXISTS prevent\_UserAccount\_UserID\_Update;~~

~~DELIMITER $$~~

~~CREATE TRIGGER prevent\_UserAccount\_UserID\_Update~~

~~BEFORE UPDATE ON USER\_ACCOUNT~~

~~FOR EACH ROW~~

~~BEGIN~~

~~IF (NEW.USER\_ID <> OLD.USER\_ID)~~

~~THEN~~

~~SIGNAL SQLSTATE '45000'~~

~~SET MESSAGE\_TEXT = 'You cannot update a USER\_ID in the USER\_ACCOUNT table.';~~

~~END IF;~~

~~END;~~

~~$$~~

~~DELIMITER ;~~

~~DROP TRIGGER IF EXISTS prevent\_SecureNote\_UserID\_Update;~~

~~DELIMITER $$~~

~~CREATE TRIGGER prevent\_SecureNote\_UserID\_Update~~

~~BEFORE UPDATE ON SECURE\_NOTE~~

~~FOR EACH ROW~~

~~BEGIN~~

~~IF (NEW.USER\_ID <> OLD.USER\_ID)~~

~~THEN~~

~~SIGNAL SQLSTATE '45000'~~

~~SET MESSAGE\_TEXT = 'You cannot update a USER\_ID in the SECURE\_NOTE table.';~~

~~END IF;~~

~~END;~~

~~$$~~

~~DELIMITER ;~~

~~DROP TRIGGER IF EXISTS prevent\_CC\_INFO\_UserID\_Update;~~

~~DELIMITER $$~~

~~CREATE TRIGGER prevent\_CC\_INFO\_UserID\_Update~~

~~BEFORE UPDATE ON CC\_INFO~~

~~FOR EACH ROW~~

~~BEGIN~~

~~IF (NEW.USER\_ID <> OLD.USER\_ID)~~

~~THEN~~

~~SIGNAL SQLSTATE '45000'~~

~~SET MESSAGE\_TEXT = 'You cannot update a USER\_ID in the CC\_INFO table.';~~

~~END IF;~~

~~END;~~

~~$$~~

~~DELIMITER ;~~

~~DROP TRIGGER IF EXISTS prevent\_NOTE\_ID\_Update;~~

~~DELIMITER $$~~

~~CREATE TRIGGER prevent\_NOTE\_ID\_Update~~

~~BEFORE UPDATE ON SECURE\_NOTE~~

~~FOR EACH ROW~~

~~BEGIN~~

~~IF (NEW.NOTE\_ID <> OLD.NOTE\_ID)~~

~~THEN~~

~~SIGNAL SQLSTATE '45000'~~

~~SET MESSAGE\_TEXT = 'You cannot update a NOTE\_ID in the SECURE\_NOTE table.';~~

~~END IF;~~

~~END;~~

~~$$~~

~~DELIMITER ;~~

~~DROP TRIGGER IF EXISTS prevent\_CC\_ID\_Update;~~

~~DELIMITER $$~~

~~CREATE TRIGGER prevent\_CC\_ID\_Update~~

~~BEFORE UPDATE ON CC\_INFO~~

~~FOR EACH ROW~~

~~BEGIN~~

~~IF (NEW.CC\_ID <> OLD.CC\_ID)~~

~~THEN~~

~~SIGNAL SQLSTATE '45000'~~

~~SET MESSAGE\_TEXT = 'You cannot update a CC\_ID in the CC\_INFO table.';~~

~~END IF;~~

~~END;~~

~~$$~~

~~DELIMITER ;~~

~~/\*trigger will add a user\_ID to the USER table if it doesn't exists\*/~~

~~DROP TRIGGER IF EXISTS add\_ID\_to\_USER\_table;~~

~~DELIMITER $$~~

~~CREATE TRIGGER add\_ID\_to\_USER\_table~~

~~BEFORE INSERT ON USER\_ACCOUNT~~

~~FOR EACH ROW~~

~~BEGIN~~

~~IF NOT EXISTS (SELECT USER\_ID FROM USER WHERE USER\_ID = NEW.USER\_ID)~~

~~THEN~~

~~INSERT INTO USER (USER\_ID)~~

~~VALUES(NEW.USER\_ID);~~

~~END IF;~~

~~END;~~

~~$$~~

~~DELIMITER ;~~