Database Challenge

Volatile Compound Sensor

Sentient

11/26/13

Contents

Business Rules……………………………………………………………………………………………………………………………………….3

ER Diagram……………………………………………………………………………………………………………………………………………4

Table Creation Scripts………………………………………………………………………………………………………………………..5-7

DCL Scripts…………………………………………………………………………………………………………………………………………8-9

DML Scripts…………………………………………………………………………………………………………………………………………10

Triggers……………………………………………………………………………………………………………………………………………….11

Example Scripts…..……………………………………………………………………………………………………………………………...12

Business Rules

* The system shall have Volatile Organic Components(VOC).
* The system shall have VOC Readings every hour.
* The system shall have Alert List Subscribers
* The system shall keep track of all the active Monitors.
* The system shall have a list of Alert Subscribers and the Monitors they are subscribed to.
* An Alert Subscriber can subscribe to many Monitors.
* A Monitor can have many Alert Subscribers
* A Monitor will not be moved once it is set up.
* An Alert Subscriber shall submit their first name, last name, and email address
* A VOC will have a name, danger zone, protection information, and symptom information
* A Monitor shall have a latitude, longitude, and zip code
* A VOC Reading shall have a Monitor, VOC, Level, Date, and Hour.
* An Alert Subscribe shall be allowed to un-subscribe to a Monitor.
* The system shall allow for VOCs to be added
* The system shall not allow for VOC Readings to be updated once they are inserted
* The system shall have integrity checks for the zip code
* The system shall have integrity checks for the email address
* The system shall have integrity checks for the VOCReading level ranges
* The system shall have integrity checks for the VOC danger zones
* The system shall allow for one Database Administrator
* The system shall allow for four Team: Sentient Database users.

Entity Relationship Diagram

An ER diagram is a visual representation of a database. The VOC Monitoring System’s database design is represented below.

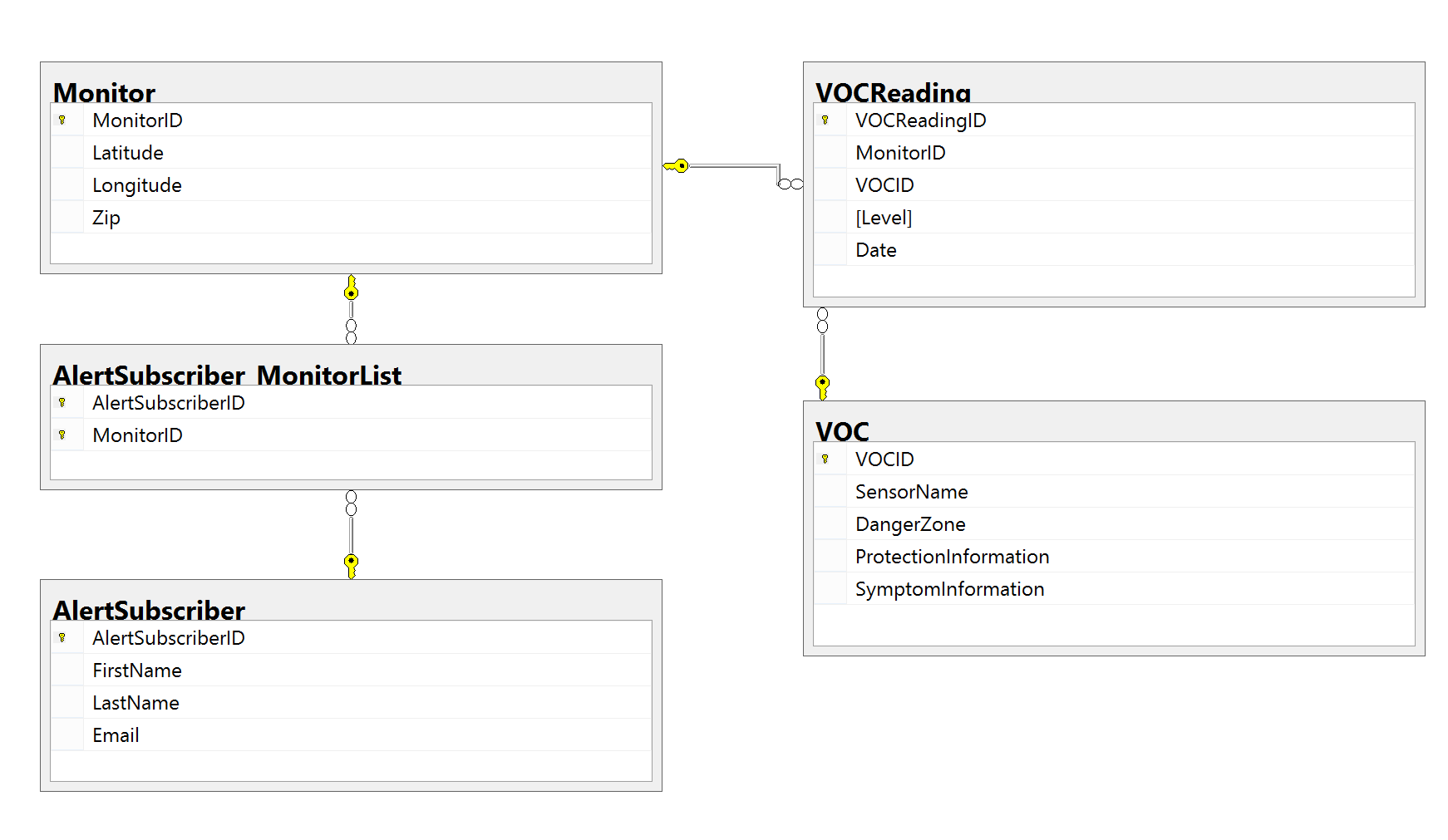


Table Creation Scripts

Following is the table creation script that initiates the database for the VOC Monitoring System. The purpose of each table and an explanation of the columns that define that table conclude this section.

Database Creation Script

IF DB\_ID('VOCMS') IS NOT NULL

DROP DATABASE VOCMS;

GO

CREATE DATABASE VOCMS;

GO

USE VOCMS;

CREATE TABLE AlertSubscriber

(

AlertSubscriberID INT CONSTRAINT Alert\_PK PRIMARY KEY IDENTITY,

FirstName VARCHAR(30) NOT NULL,

LastName VARCHAR(30) NOT NULL,

DateAdded DateTime DEFAULT GetDate(),

Email VARCHAR(50) NOT NULL,

CONSTRAINT Email\_check CHECK( Email LIKE '%@%.%' )

);

CREATE TABLE Monitor

(

MonitorID INT CONSTRAINT Monitor\_PK PRIMARY KEY IDENTITY,

Latitude float NOT NULL,

Longitude DECIMAL(4,4) NOT NULL,

Zip VARCHAR(5) NOT NULL,

CONSTRAINT Zip\_Check CHECK( Zip LIKE '[0-9][0-9][0-9][0-9][0-9]')

--Zip Codes are five digits

);

CREATE TABLE AlertSubscriber\_MonitorList

(

AlertSubscriberID INT CONSTRAINT Alert\_FK REFERENCES AlertSubscriber(AlertSubscriberID),

MonitorID INT CONSTRAINT Alert\_FK REFERENCES Monitor(MonitorID),

CONSTRAINT AlertList\_PrimaryKey PRIMARY KEY ( AlertSubscriberID, MonitorID )

);

CREATE TABLE VOC

(

VOCID INT CONSTRAINT VOC\_PK PRIMARY KEY IDENTITY,

SensorName VARCHAR(30) NOT NULL,

DangerZone INT NOT NULL,

ProtectionInformation VARCHAR(2048) NOT NULL,

SymptomInformation VARCHAR(2048) NOT NULL,

CONSTRAINT DangerZone\_Check CHECK( DangerZone BETWEEN 50 AND 5000)

--Precision of sensor falls between 50 - 5000ppm

);

CREATE TABLE VOCReading

(

VOCReadingID INT CONSTRAINT VOCReading\_PK PRIMARY KEY IDENTITY,

MonitorID INT CONSTRAINT VOCReading\_Monitor\_FK REFERENCES Monitor(MonitorID),

VOCID INT CONSTRAINT VOCReading\_VOC\_FK REFERENCES VOC(VOCID),

Level INT NOT NULL,

Date DateTime NOT NULL,

--Not when added to database but when retrieved through sensor

CONSTRAINT VOC\_Level\_Check CHECK( Level BETWEEN 50 AND 5000 )

--Precision of sensor falls between 50 - 5000ppm

);

Create Tables Defined

AlertSubscriber:

The Alert Subscriber table is used to insert and retrieve subscriber data in order to create an e-mail alert system. The table is referenced in the alert list in order to connect each user to a specific monitor. There is a check on e-mail as an attempt to ensure that a valid e-mail was presented for a user. Name data is retrieved in order to personalize e-mails.

CREATE TABLE AlertSubscriber

(

AlertSubscriberID INT CONSTRAINT Alert\_PK PRIMARY KEY IDENTITY,

FirstName VARCHAR(30) NOT NULL,

LastName VARCHAR(30) NOT NULL,

DateAdded DateTime DEFAULT GetDate(),

Email VARCHAR(50) NOT NULL,

CONSTRAINT Email\_check CHECK( Email LIKE '%@%.%' )

);

Monitor:

The Monitor table is used to record the existence and location of monitors. The Monitor table is referenced in the AlertSubscriber\_MonitorList in order to connect users to each monitor. The Zip code is recorded to simplify connecting users to monitors. The longitude and latitude is obtained for location on a Google’s map. A check ensures that the inserted zip code is five numbers.

CREATE TABLE Monitor

(

MonitorID INT CONSTRAINT Monitor\_PK PRIMARY KEY IDENTITY,

Latitude float NOT NULL,

Longitude DECIMAL(4,4) NOT NULL,

Zip VARCHAR(5) NOT NULL,

CONSTRAINT Zip\_Check CHECK( Zip LIKE '[0-9][0-9][0-9][0-9][0-9]')

--Zip Codes are five digits

);

AlertSubscriber\_MonitorList:

The AlertSubscriber\_MonitorList table is an intermediate table used to connect Alert Subscribers to Monitors. This connection is required in order to send an e-mail to the appropriate user when a monitor records a value that is above the danger zone for their area.

CREATE TABLE AlertSubscriber\_MonitorList

(

AlertSubscriberID INT CONSTRAINT Alert\_FK REFERENCES AlertSubscriber(AlertSubscriberID),

MonitorID INT CONSTRAINT Alert\_FK REFERENCES Monitor(MonitorID),

CONSTRAINT AlertList\_PrimaryKey PRIMARY KEY ( AlertSubscriberID, MonitorID )

);

VOC:

The VOC table records the volatile organic compound that each monitor sensor accounts for. The VOCID’s are referenced in the VOCReading table in order to tie each value recorded by the monitor to that specific VOC. Protection and Symptom information is used in the Alert System e-mails and the monitor Web Application.

CREATE TABLE VOC

(

VOCID INT CONSTRAINT VOC\_PK PRIMARY KEY IDENTITY,

SensorName VARCHAR(30) NOT NULL,

DangerZone INT NOT NULL,

ProtectionInformation VARCHAR(2048) NOT NULL,

SymptomInformation VARCHAR(2048) NOT NULL,

CONSTRAINT DangerZone\_Check CHECK( DangerZone BETWEEN 50 AND 5000)

--Precision of sensor falls between 50 - 5000ppm

);

VOCReading:

The VOCRecords table is used to persist the volatile organic compounds retrieved by the monitoring system. With each level the date and time it was retrieved is recorded for the graphs on the web application. Each retrieved represents a volatile organic compound (VOCID) and is retrieved from a specific monitor (MonitorID).

CREATE TABLE VOCReading

(

VOCReadingID INT CONSTRAINT VOCReading\_PK PRIMARY KEY IDENTITY,

MonitorID INT CONSTRAINT VOCReading\_Monitor\_FK REFERENCES

Monitor(MonitorID),

VOCID INT CONSTRAINT VOCReading\_VOC\_FK REFERENCES VOC(VOCID),

Level INT NOT NULL,

Date DateTime NOT NULL,

CONSTRAINT VOC\_Level\_Check CHECK( Level BETWEEN 50 AND 5000 )

--Precision of sensor falls between 50 - 5000ppm

);

Data Control Language Scripts

DCL scripts are used to control the permissions given to users and logins of the database. All standard SQL permissions are granted to all Sentient team members except ALTER, UPDATE, and INSERT for the VOCReading table. The VOCReading should only allow insertions of VOC data from the admin.

Roles

There are three Role in the Monitor System: Sentient member whom can alter monitor, sensor, and altertlist tables and data, and an Administrator that runs scripts to update VOC level data.

USE VOCMS;

Go

CREATE ROLE Sentient\_Member

GRANT SELECT, EXECUTE, UPDATE, INSERT, DELETE, ALTER

ON DATABASE::VOCMS

TO Sentient\_Member

REVOKE UPDATE, INSERT, ALTER, DELETE

ON VOCReading

TO Sentient\_Member

--Sentient Members cannot make changes to the VOCReading table

USE VOCMS;

Go

CREATE ROLE Administrator

GRANT SELECT, EXECUTE, UPDATE, INSERT, DELETE, ALTER

ON DATABASE::VOCMS

TO Administrator --Administrator has complete access to database

Logins

There is a single administrator that controls the database and four Sentient members.

GO

CREATE LOGIN Administrator WITH PASSWORD = 'Document is perfect';

Go

CREATE LOGIN Thyanna WITH PASSWORD = 'This';

GO

CREATE LOGIN Dylan WITH PASSWORD = 'Is';

GO

CREATE LOGIN Charlie WITH PASSWORD = 'Subliminal';

GO

CREATE LOGIN Jeff WITH PASSWORD = 'Messaging';

Users

CREATE USER Thyanna

FOR LOGIN Thyanna

CREATE USER Charlie

FOR LOGIN Charlie

CREATE USER Dylan

FOR LOGIN Dylan

CREATE USER Jeff

FOR LOGIN Jeff

Alter ROLEs

Roles are then altered in order to add the respective roles.

ALTER ROLE Sentient\_Member ADD MEMBER Thyanna

ALTER ROLE Sentient\_Member ADD MEMBER Dylan

ALTER ROLE Sentient\_Member ADD MEMBER Jeff

ALTER ROLE Sentient\_Member ADD MEMBER Charlie

ALTER ROLE Administrator ADD MEMBER Administrator

DML Scripts

Data Manipulation Language scripts are used to INSERT, UPDATE, and DELETE data from the database. The following are examples for succeeding and failing DML scripts.

Succeeding Scripts

INSERT success

INSERT INTO VOCReading ( MonitorID, VOCID, Level, Date ) VALUES

(1, 5, 550, '2013-06-02 04:03:55' )

UPDATE success

UPDATE Sensor

SET DangerZone = 173.0

WHERE VOCName = 'Ammonia';

Delete success

DELETE FROM AlertSubscriber

WHERE SubscriberID = (SELECT SubscriberID

FROM Subscriber

WHERE FirstName = 'Thyanna' AND LastName = 'Voisine')

DELETE FROM Subscriber

WHERE FirstName = 'Thyanna' AND LastName = 'Voisine'

Failing Scripts

DELETE fail

DELETE FROM VOCReading

WHERE VOCID = (SELECT VOCID

FROM VOC

WHERE VOCName = 'Methyl mercaptan')

--Fails due to permissions. No one is allowed to delete data from VOCLevels.

UPDATE fail

UPDATE VOCReading

SET Level = 1

WHERE VOCID = 2

-- Fails due to not having permissions. No one is allowed to UPDATE VOCLevels.

Insert fail

INSERT INTO Monitor ( Latitude, Longitude, Zip ) VALUES

( 42.254173, -121.786324, 976013 )

--Fails because zip codes must be 5 characters long.

Triggers

A Trigger is used to ensure that if a specific script is executed, that all other scripts that must also be executed, are run. A Trigger can also be used to replace a scripts. The following scripts ensures that upon entering a Sensor, the first character of that VOCs Name is capitalized.

CREATE TRIGGER VOC\_INSERT\_UPDATE

ON VOC

AFTER INSERT, UPDATE

AS

UPDATE VOC

SET VOCName = UPPER(LEFT(VOCName,1)) +

RIGHT(VOCName, LEN( VOCName)-1);

--The purpose of this trigger is to make all sensor names that are entered to have the ----first character be capital.

CREATE TRIGGER Subscriber\_INSERT

ON AlertSubscriber

AFTER INSERT

AS

IF ((SELECT Email

FROM INSERTED ) IN ( SELECT Email

FROM AlertSubscriber

WHERE datepart(hh, DateAdded) = datepart( hh, GetDate() ) ) )

BEGIN

DELETE FROM ALERTSubscriber

WHERE Email = (SELECT Email

FROM INSERTED ) AND

datepart(hh, DateAdded) = datepart(hh, GetDate())

END

--The purpose of this trigger is to ensure duplicate e-mail address are not entered into –the database

Example Scripts

The following are examples of procedures that will be regularly executed for the VOC monitoring system.

Procedure 1:

This procedure returns all VOC levels that have been stored for a specific VOC. This procedure will be used in order to update the VOC (ppm) VS. Time (hours) graph on the VOC Monitor Website. When an end user chooses to see Formaldehyde on the graph, this procedure will be passed ‘Formaldehyde’ and return all the VOC levels along with the time each was retrieved.

USE VOCMS;

if OBJECT\_ID('GetVOCData') IS NOT NULL

DROP PROC GetVOCData

GO

CREATE PROC GetVOCData

@SensorNameVar varchar(30)

AS

SELECT Date,

Time,

Sensor.SensorName,

VOCLevels.Level

FROM VOCLevels JOIN Sensor

ON VOCLevels.SensorID = Sensor.SensorID

WHERE Sensor.SensorName = @SensorNameVar

ORDER BY Date, Time

Procedure 2:

This procedure is used to retrieve the e-mail address of end users. Specifically this procedure checks VOC levels against their respective danger zone value. If a value proves to be too high for human safety, the e-mails of all users that are registered to that monitor are returned. Along with the VOC that caused the alert, its respective symptom and protection information is returned.

CREATE PROC CheckDangerLevel

AS

SELECT DISTINCT Email,

Sensor.SensorName,

ProtectionInformation

FROM Customer JOIN AlertList ON Customer.CustomerID = AlertList.CustomerID

JOIN Monitor ON AlertList.MonitorID = Monitor.MonitorID

JOIN VOCLevels ON Monitor.MonitorID = VOCLevels.MonitorID

JOIN Sensor ON VOCLevels.SensorID = Sensor.SensorID

WHERE Monitor.Zip =

(SELECT Monitor.Zip

FROM Monitor JOIN VOCLevels ON Monitor.MonitorID = VOCLevels.MonitorID JOIN Sensor ON VOCLevels.SensorID = Sensor.SensorID

WHERE VOCLevels.Level >= Sensor.DangerZone )