PRACTICO:

* Crear una tabla 'tanques' que contenga:
  + nombre\_tanque VARCHAR(45) PK
  + nivel\_umbral\_delta DOUBLE
  + temperatura\_umbral\_delta DOUBLE
  + nivel\_umbral\_delta\_segundos INT
  + temperatura\_umbral\_delta\_segundos INT
* Modificar tabla 'ultimos\_estados' para que contenga la FK a 'tanques'
* Crear tabla 'alarmas\_tanques'
  + id BIGINT (PK AI)
  + fecha\_hora DATETIME
  + tanque VARCHAR(45)
  + tipo VARCHAR(20) Tipos posibles: 'nivel' o 'temperatura' (controlar y rechazar si los valores no son correctos)
  + delta DOUBLE
  + dif\_secs BIGINT
  + informado` TINYINT(4) NOT NULL DEFAULT 0
* Crear un evento que se ejecute cada 10 segundos y que realice la siguiente tarea:
  + Verifique valores delta de temperatura y nivel por tanque en 'ultimos\_estados'
  + Si cualquier delta supera nivel\_umbral\_delta o temperatura\_umbral\_delta, según corresponda y además se supera la 'dif\_secs' en función de 'nivel\_umbral\_delta\_segundos' o 'temperatura\_umbral\_delta\_segundos' se debe insertar una fila en la tabla 'alarmas\_tanques' con los datos correspondientes. Para realizar este cálculo debe crear una función
* Crear un procedimiento almacenado que permita ingresar el nivel y la temperatura de un tanque (en la tabla 'ultimos\_estados', sólo debe recibir como parámetros tanque, nivel y temperatura y siempre establecer el último estado. Además anotará en una nueva tabla de valores históricos 'valores\_historicos': tanque, fecha\_hora, nivel y temperatura, pero solo a intervalos mínimos, medidos en segundos, establecidos en un nuevo campo por tanque denominado 'intervalo\_minimo\_de\_almacenamiento', por ejemplo:

Intervalo=30 segundos ('intervalo\_minimo\_de\_almacenamiento'=30)

Valores recibidos:

* Tanque, Fecha Hora, Nivel, Temperatura, Histórico
* 'TK 1', '2021-1-1 00:00:00', 400, 23 **Si**
* 'TK 1', '2021-1-1 00:00:11', 401, 23 No
* 'TK 1', '2021-1-1 00:00:21', 402, 24 No
* 'TK 1', '2021-1-1 00:00:32', 403, 25 **Si**
* 'TK 1', '2021-1-1 00:00:45', 404, 24 No
* 'TK 1', '2021-1-1 00:00:56', 405, 23 No
* 'TK 1', '2021-1-1 00:01:06', 406, 22 **Si**
* 'TK 1', '2021-1-1 00:01:16', 407, 21 No
* Crear evento que llame al procedimiento almacenado anterior a intervalos de 7 segundos enviando datos aleatorios de nivel y temperatura (en un rango dado TIP: crear una función!!!!!) para un tanque en particular

------------------------------------------------------------

show processlist;

set global event\_scheduler=on;

delimiter $$

CREATE EVENT [IF NOT EXISTS] nombre

ON SCHEDULE definicion\_del\_schedule

DO

BEGIN

sentencia1;

sentencia1;

sentencia1;

END$$

delimiter ;

CREATE EVENT evento\_1

ON SCHEDULE AT CURRENT\_TIMESTAMP + INTERVAL 15 SECOND

DO

INSERT INTO eventos (texto, fecha\_hora) VALUES (LEFT(MD5(RAND()),8), NOW());

SELECT \* FROM eventos;

CREATE EVENT evento\_2

ON SCHEDULE

EVERY 10 SECOND

STARTS CURRENT\_TIMESTAMP

ENDS CURRENT\_TIMESTAMP + INTERVAL 40 second

DO

INSERT INTO eventos (texto, fecha\_hora) VALUES (LEFT(MD5(RAND()),8), NOW());

DROP TABLE IF EXISTS ultimos\_estados;

CREATE TABLE `ultimos\_estados` (

`id` INT NOT NULL AUTO\_INCREMENT,

`fecha\_hora` DATETIME NULL,

`tanque` VARCHAR(45) NOT NULL,

`nivel` DOUBLE NOT NULL DEFAULT 0,

`temperatura` DOUBLE NOT NULL DEFAULT 0,

`nivel\_delta` DOUBLE NOT NULL DEFAULT 0,

`temperatura\_delta` DOUBLE NOT NULL DEFAULT 0,

`dif\_secs` BIGINT NOT NULL DEFAULT 0,

PRIMARY KEY (`id`),

UNIQUE INDEX `tanque\_UNIQUE` (`tanque` ASC));

DELIMITER $$

DROP TRIGGER IF EXISTS `actualiza\_deltas\_ultimos\_estados`$$

CREATE TRIGGER `actualiza\_deltas\_ultimos\_estados` BEFORE UPDATE ON ultimos\_estados

FOR EACH ROW

BEGIN

SET NEW.nivel\_delta=NEW.nivel-OLD.nivel;

SET NEW.temperatura\_delta=NEW.temperatura-OLD.temperatura;

SET NEW.dif\_secs=TIMESTAMPDIFF(SECOND,OLD.fecha\_hora,NOW());

END$$

DELIMITER ;

INSERT INTO `ultimos\_estados` (`fecha\_hora`, `tanque`, `nivel`, `temperatura`) VALUES ('2021-8-19 14:01:01', 'TK1', '100.01', '23.4');

INSERT INTO `ultimos\_estados` (`fecha\_hora`, `tanque`, `nivel`, `temperatura`) VALUES ('2021-8-19 14:01:02', 'TK2', '500.5', '22.3');

UPDATE `ultimos\_estados` SET fecha\_hora=now(), `nivel` = '401.11', `temperatura` = '23.8' WHERE (`id` = '1');

UPDATE `ultimos\_estados` SET fecha\_hora=now(), `nivel` = '801.17', `temperatura` = '24.5' WHERE (`id` = '2');

DELIMITER $$

DROP FUNCTION IF EXISTS existeColumna $$

CREATE FUNCTION existeColumna (tabla VARCHAR(100), columna VARCHAR(100))

RETURNS INT

RETURN (

SELECT COUNT(COLUMN\_NAME)

FROM INFORMATION\_SCHEMA.columns

WHERE TABLE\_SCHEMA = DATABASE()

AND TABLE\_NAME = tabla

AND COLUMN\_NAME = columna

)$$

SELECT COUNT(COLUMN\_NAME)

FROM INFORMATION\_SCHEMA.columns

WHERE TABLE\_SCHEMA = 'prueba\_federacion'

AND TABLE\_NAME = 'eventos'

AND COLUMN\_NAME = 'id';

select

# - Ahorran ancho de banda

# - Centralizan lógica de negocios

# - Aumentan la performance y respuesta general

DELIMITER $$

DROP PROCEDURE IF EXISTS agregaColumnaSiNoExiste $$

CREATE PROCEDURE agregaColumnaSiNoExiste (

IN tabla VARCHAR(100)

, IN columna VARCHAR(100)

, IN defColumna VARCHAR(100)

)

BEGIN

SET @existe = existeColumna(tabla, columna);

IF (@existe = 0) THEN

SET @ddl = CONCAT('ALTER TABLE ', tabla);

SET @ddl = CONCAT(@ddl, ' ', 'ADD COLUMN') ;

SET @ddl = CONCAT(@ddl, ' ', columna);

SET @ddl = CONCAT(@ddl, ' ', defColumna);

PREPARE stmt FROM @ddl;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

END IF;

END$$

DELIMITER ;

CALL agregaColumnaSiNoExiste('ultimos\_estados','fecha\_hora','varchar(10)');

CALL agregaColumnaSiNoExiste('ultimos\_estados','densidad',"double not null default '0'");