**Granja Avícola – Módulo de Alimentos  
Stored Procedures y Triggers SQL**

# Tabla de Contenido

1. Notas generales

2. Orden de creación y dependencias

3. Stored Procedures (SP)

3.1 sp\_material\_upsert

3.2 sp\_formula\_base\_create

3.3 sp\_formula\_base\_detalle\_upsert

3.4 sp\_formula\_cond\_create

3.5 sp\_formula\_cond\_detalle\_upsert

3.6 sp\_paramnutricional\_upsert

3.7 sp\_get\_formula\_vigente

3.8 sp\_lote\_formula\_assign

3.9 sp\_calc\_requerimiento

4. Triggers

4.1 trg\_material\_ai (auditoría insert)

4.2 trg\_material\_au (auditoría update + notificación cambio precio)

4.3 trg\_formulaalimento\_bi (anti-solape)

4.4 trg\_formulacondicion\_bi (anti-solape)

4.5 trg\_lote\_formula\_bi (auto-cierre previo)

**B.1 SP: Insert / Update Material + Histórico**

**Nombre:** sp\_material\_upsert

DELIMITER $$

CREATE PROCEDURE sp\_material\_upsert (

IN p\_material\_id INT,

IN p\_nombre VARCHAR(100),

IN p\_peso\_kg DECIMAL(10,4),

IN p\_unidad\_id INT,

IN p\_activo TINYINT,

IN p\_usuario\_id INT,

IN p\_fecha\_inicio\_hist DATE

)

BEGIN

DECLARE v\_id INT;

IF p\_material\_id IS NULL THEN

-- Insertar nuevo material

INSERT INTO material (nombre, peso\_kg, unidad\_id, activo, creado\_por)

VALUES (p\_nombre, p\_peso\_kg, p\_unidad\_id, p\_activo, p\_usuario\_id);

SET v\_id = LAST\_INSERT\_ID();

ELSE

-- Actualizar material existente

UPDATE material

SET nombre = p\_nombre,

peso\_kg = p\_peso\_kg,

unidad\_id = p\_unidad\_id,

activo = p\_activo,

modificado\_por = p\_usuario\_id,

modificado\_fecha = NOW()

WHERE id = p\_material\_id;

SET v\_id = p\_material\_id;

END IF;

-- Registrar auditoría

INSERT INTO audit\_log (tabla, pk\_valor, accion, usuario\_id, detalle)

VALUES ('material', v\_id, IF(p\_material\_id IS NULL, 'INSERT', 'UPDATE'), p\_usuario\_id, NULL);

END$$

DELIMITER ;

**B.2 SP: Crear Nueva Fórmula Base (versionando)**

**Reglas:**

DELIMITER $$

CREATE PROCEDURE sp\_formula\_base\_create (

IN p\_tipo\_ave\_id INT,

IN p\_etapa\_id INT,

IN p\_fecha\_inicio DATE,

IN p\_observaciones TEXT,

IN p\_usuario\_id INT,

OUT p\_new\_formula\_id INT

)

BEGIN

UPDATE formulaalimento

SET fecha\_fin = DATE\_SUB(p\_fecha\_inicio, INTERVAL 1 DAY),

modificado\_por = p\_usuario\_id,

modificado\_fecha = NOW()

WHERE tipo\_ave\_id = p\_tipo\_ave\_id

AND etapa\_id = p\_etapa\_id

AND fecha\_fin IS NULL;

-- Crea nueva fórmula

INSERT INTO formulaalimento

(tipo\_ave\_id, etapa\_id, fecha\_inicio, observaciones, creado\_por)

VALUES

(p\_tipo\_ave\_id, p\_etapa\_id, p\_fecha\_inicio, p\_observaciones, p\_usuario\_id);

SET p\_new\_formula\_id = LAST\_INSERT\_ID();

INSERT INTO audit\_log (tabla, pk\_valor, accion, usuario\_id, detalle)

VALUES ('formulaalimento', p\_new\_formula\_id, 'INSERT', p\_usuario\_id, NULL);

END$$

DELIMITER ;

**B.3 SP: Agregar/Actualizar Detalle de Fórmula Base**

DELIMITER $$

CREATE PROCEDURE sp\_formula\_base\_detalle\_upsert (

IN p\_formula\_id INT,

IN p\_material\_id INT,

IN p\_porcentaje DECIMAL(5,2),

IN p\_usuario\_id INT

)

BEGIN

INSERT INTO detalleformula (formula\_id, material\_id, porcentaje)

VALUES (p\_formula\_id, p\_material\_id, p\_porcentaje)

ON DUPLICATE KEY UPDATE porcentaje = VALUES(porcentaje);

INSERT INTO audit\_log (tabla, pk\_valor, accion, usuario\_id, detalle)

VALUES ('detalleformula', CONCAT(p\_formula\_id,':',p\_material\_id), 'UPDATE', p\_usuario\_id, NULL);

END$$

DELIMITER ;

**B.4 SP: Crear Fórmula Condicionada ()**

DELIMITER $$

CREATE PROCEDURE sp\_formula\_cond\_create (

IN p\_formula\_base\_id INT,

IN p\_condicion\_id INT,

IN p\_fecha\_inicio DATE,

IN p\_observaciones TEXT,

IN p\_usuario\_id INT,

OUT p\_new\_formula\_cond\_id INT

)

BEGIN

-- Cierra vigente previa para misma base+condición

UPDATE formulacondicion

SET fecha\_fin = DATE\_SUB(p\_fecha\_inicio, INTERVAL 1 DAY),

modificado\_por = p\_usuario\_id,

modificado\_fecha = NOW()

WHERE formula\_base\_id = p\_formula\_base\_id

AND condicion\_id = p\_condicion\_id

AND fecha\_fin IS NULL;

INSERT INTO formulacondicion

(formula\_base\_id, condicion\_id, fecha\_inicio, observaciones, creado\_por)

VALUES

(p\_formula\_base\_id, p\_condicion\_id, p\_fecha\_inicio, p\_observaciones, p\_usuario\_id);

SET p\_new\_formula\_cond\_id = LAST\_INSERT\_ID();

INSERT INTO audit\_log (tabla, pk\_valor, accion, usuario\_id, detalle)

VALUES ('formulacondicion', p\_new\_formula\_cond\_id, 'INSERT', p\_usuario\_id, NULL);

END$$

DELIMITER ;

**B.5 SP: Obtener Fórmula Vigente**

DELIMITER $$

CREATE PROCEDURE sp\_get\_formula\_vigente (

IN p\_tipo\_ave\_id INT,

IN p\_etapa\_id INT,

IN p\_condicion\_id INT, -- NULL si no aplica

IN p\_fecha DATE,

OUT p\_formula\_tipo ENUM('BASE','CONDICIONADA'),

OUT p\_formula\_id INT

)

BEGIN

DECLARE v\_base\_id INT;

DECLARE v\_cond\_id INT;

-- Base vigente

SELECT id INTO v\_base\_id

FROM formulaalimento

WHERE tipo\_ave\_id = p\_tipo\_ave\_id

AND etapa\_id = p\_etapa\_id

AND p\_fecha BETWEEN fecha\_inicio AND IFNULL(fecha\_fin,'9999-12-31')

ORDER BY fecha\_inicio DESC

LIMIT 1;

IF p\_condicion\_id IS NOT NULL THEN

SELECT id INTO v\_cond\_id

FROM formulacondicion

WHERE formula\_base\_id = v\_base\_id

AND condicion\_id = p\_condicion\_id

AND p\_fecha BETWEEN fecha\_inicio AND IFNULL(fecha\_fin,'9999-12-31')

ORDER BY fecha\_inicio DESC

LIMIT 1;

END IF;

IF v\_cond\_id IS NOT NULL THEN

SET p\_formula\_tipo = 'CONDICIONADA';

SET p\_formula\_id = v\_cond\_id;

ELSE

SET p\_formula\_tipo = 'BASE';

SET p\_formula\_id = v\_base\_id;

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE sp\_lote\_formula\_assign (

IN p\_lote\_id INT,

IN p\_formulaalimento\_id INT,

IN p\_formulacondicion\_id INT,

IN p\_fecha\_inicio DATE,

IN p\_observaciones TEXT,

IN p\_usuario\_id INT,

OUT p\_new\_lote\_formula\_id INT

)

BEGIN

-- Cierra asignación previa (cualquiera)

UPDATE lote\_formula

SET fecha\_fin = DATE\_SUB(p\_fecha\_inicio, INTERVAL 1 DAY)

WHERE lote\_id = p\_lote\_id

AND fecha\_fin IS NULL;

INSERT INTO lote\_formula

(lote\_id, formulaalimento\_id, formulacondicion\_id, fecha\_inicio, observaciones, aplicado\_por)

VALUES

(p\_lote\_id, p\_formulaalimento\_id, p\_formulacondicion\_id, p\_fecha\_inicio, p\_observaciones, p\_usuario\_id);

SET p\_new\_lote\_formula\_id = LAST\_INSERT\_ID();

INSERT INTO audit\_log (tabla, pk\_valor, accion, usuario\_id, detalle)

VALUES ('lote\_formula', p\_new\_lote\_formula\_id, 'INSERT', p\_usuario\_id, NULL);

END$$

DELIMITER ;

**B.7 SP: Calcular Requerimiento Total (kg + costo)**

DELIMITER $$

CREATE PROCEDURE sp\_calc\_requerimiento (

IN p\_formula\_tipo ENUM('BASE','CONDICIONADA'),

IN p\_formula\_id INT,

IN p\_cant\_aves INT,

IN p\_consumo\_g\_dia DECIMAL(10,2),

IN p\_dias INT

)

BEGIN

/\*

Total alimento (kg) = (cant\_aves \* consumo\_g\_dia \* dias) / 1000

\*/

SELECT

m.id AS material\_id,

m.nombre,

d.porcentaje,

ROUND(((p\_cant\_aves \* p\_consumo\_g\_dia \* p\_dias)/1000) \* (d.porcentaje/100), 2) AS kg\_requeridos,

m.precio\_kg,

ROUND((((p\_cant\_aves \* p\_consumo\_g\_dia \* p\_dias)/1000) \* (d.porcentaje/100)) \* m.precio\_kg, 2) AS costo

FROM (

SELECT material\_id, porcentaje

FROM detalleformula

WHERE p\_formula\_tipo = 'BASE' AND formula\_id = p\_formula\_id

UNION ALL

SELECT material\_id, porcentaje

FROM detalleformulacondicion

WHERE p\_formula\_tipo = 'CONDICIONADA' AND formula\_cond\_id = p\_formula\_id

) d

JOIN material m ON m.id = d.material\_id;

END$$

DELIMITER ;

**Triggers**

**C.1 Auditoría genérica (ejemplo: MATERIAL)**

DELIMITER $$

CREATE TRIGGER trg\_material\_ai

AFTER INSERT ON material

FOR EACH ROW

BEGIN

INSERT INTO audit\_log (tabla, pk\_valor, accion, usuario\_id, detalle)

VALUES ('material', NEW.id, 'INSERT', NEW.creado\_por, NULL);

END$$

CREATE TRIGGER trg\_material\_au

AFTER UPDATE ON material

FOR EACH ROW

BEGIN

INSERT INTO audit\_log (tabla, pk\_valor, accion, usuario\_id, detalle)

VALUES ('material', NEW.id, 'UPDATE', NEW.modificado\_por, JSON\_OBJECT(

'old\_precio', OLD.precio\_kg, 'new\_precio', NEW.precio\_kg,

'old\_proteina', OLD.proteina, 'new\_proteina', NEW.proteina

));

END$$

DELIMITER ;

**C.2 Registrar histórico al cambiar Material**

DELIMITER $$

CREATE TRIGGER trg\_material\_hist\_bu

BEFORE UPDATE ON material

FOR EACH ROW

BEGIN

INSERT INTO material\_hist (

material\_id, fecha\_inicio, proteina, energia, fibra, calcio, fosforo, precio\_kg, observaciones, creado\_por

)

VALUES (

OLD.id, CURDATE(), OLD.proteina, OLD.energia, OLD.fibra, OLD.calcio, OLD.fosforo, OLD.precio\_kg,

'Snapshot automático antes de actualización.', NEW.modificado\_por

);

END$$

DELIMITER ;

**C.3 Anti-solape de vigencias**

DELIMITER $$

CREATE TRIGGER trg\_formulaalimento\_bi

BEFORE INSERT ON formulaalimento

FOR EACH ROW

BEGIN

IF EXISTS (

SELECT 1 FROM formulaalimento

WHERE tipo\_ave\_id = NEW.tipo\_ave\_id

AND etapa\_id = NEW.etapa\_id

AND NEW.fecha\_inicio BETWEEN fecha\_inicio AND IFNULL(fecha\_fin,'9999-12-31')

) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Ya existe una fórmula vigente o solapada para este tipo de ave y etapa.';

END IF;

END$$

DELIMITER ;

**C.5 Notificación precio material subió > X% ()**

DELIMITER $$

CREATE TRIGGER trg\_material\_precio\_au

AFTER UPDATE ON material

FOR EACH ROW

BEGIN

IF OLD.precio\_kg > 0 AND ((NEW.precio\_kg - OLD.precio\_kg)/OLD.precio\_kg) >= 0.10 THEN

INSERT INTO notif\_queue (notif\_tipo\_id, ref\_tabla, ref\_id, mensaje)

VALUES (

1, -- asume notif\_tipo\_id=1 = "Cambio de Precio Material"

'material', NEW.id,

CONCAT('Precio de ', NEW.nombre, ' subió de ', OLD.precio\_kg, ' a ', NEW.precio\_kg)

);

END IF;

END$$

DELIMITER ;