

## Acréscimos à Base de Dados Relativos à Nota de Falta

### Novas Tabelas

Tabelas		
Ordem	Tabela	Script
1	nota_faltas	<pre> DROP TABLE IF EXISTS `nota_faltas`; CREATE TABLE IF NOT EXISTS `nota_faltas` (   `id` int(11) NOT NULL AUTO_INCREMENT,   `data` timestamp NULL DEFAULT CURRENT_TIMESTAMP,   `user_id` int(11) NOT NULL,   `cliente_id` int(11) NOT NULL,   `valor_total` decimal(8,2) DEFAULT NULL,   `iva` decimal(8,2) NOT NULL,   `valor_iva` decimal(8,2) NOT NULL,   `motivo_iva_id` int(11) DEFAULT NULL,   `aplicacao_motivo_iva` tinyint(1) NOT NULL,   `saida_id` int(11) NOT NULL,   PRIMARY KEY (`id`),   UNIQUE KEY `saida_id_UNIQUE` (`saida_id`),   KEY `fk_notas_falta_clientes1_idx` (`cliente_id`),   KEY `fk_notas_falta_user1_idx` (`user_id`),   KEY `fk_notas_falta_motivo_ivas1_idx` (`motivo_iva_id`),   KEY `fk_nota_faltas_saidas1_idx` (`saida_id`),   CONSTRAINT `fk_nota_faltas_clientes1` FOREIGN KEY (`cliente_id`) REFERENCES `clientes` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,   CONSTRAINT `fk_nota_faltas_motivo_ivas1` FOREIGN KEY (`motivo_iva_id`) REFERENCES `motivo_ivas` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,   CONSTRAINT `fk_nota_faltas_saidas1` FOREIGN KEY (`saida_id`) REFERENCES `saidas` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,   CONSTRAINT `fk_nota_faltas_users1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION ) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=latin1 COMMENT='Unique para a saida_id porque se considera um relacionamento de um para um, desta forma, a saida deve aparecer uma unica vez para cada nota_falta.'</pre>
2	iten_nota_faltas	<pre> DROP TABLE IF EXISTS `iten_nota_faltas`; CREATE TABLE IF NOT EXISTS `iten_nota_faltas`</pre>

		<pre>( `id` int(11) NOT NULL AUTO_INCREMENT, `produto_id` int(11) NOT NULL, `quantidade` int(11) NOT NULL, `valor` decimal(8,2) NOT NULL, `desconto` int(11) NOT NULL, `subtotal` decimal(8,2) NOT NULL, `nota_falta_id` int(11) NOT NULL, PRIMARY KEY (`id`), KEY `fk_iten_nota_faltas_nota_faltas1_idx` (`nota_falta_id`), CONSTRAINT `fk_iten_nota_faltas_nota_faltas1` FOREIGN KEY (`nota_falta_id`) REFERENCES `nota_faltas` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION ) ENGINE=InnoDB AUTO_INCREMENT=38 DEFAULT CHARSET=latin1</pre>
--	--	---

## Novos Procedimentos

Novos Procedimentos		
Ordem	Nome do Procedimento	Script
1	SP_decrementar_valor_total_nota_falta	<pre>DELIMITER \$\$ DROP PROCEDURE IF EXISTS `SP_decrementar_valor_total_nota_falta`\$\$ CREATE PROCEDURE `SP_decrementar_valor_total_nota_falta`(`valor` DECIMAL(8,2), `nota_falta_id` INT) BEGIN     DECLARE valor_iva DECIMAL(8,2);     DECLARE iva DECIMAL(8,2);      SET @valor_iva = (valor+(valor*17)/100);     SET @iva = ((valor*17)/100);      UPDATE nota_faltas SET nota_faltas.valor_total = (nota_faltas.valor_total - valor),     nota_faltas.iva = (nota_faltas.iva - @iva),     nota_faltas.valor_iva = (nota_faltas.valor_iva - @valor_iva) WHERE nota_faltas.id = nota_falta_id; END\$\$ DELIMITER ;</pre>
2	SP_incrementar_valor_total_nota_falta	<pre>DELIMITER \$\$</pre>

		<pre> DROP PROCEDURE IF EXISTS `SP_incrementar_valor_total_nota_falta`\$\$ CREATE PROCEDURE `SP_incrementar_valor_total_nota_falta`(`valor` DECIMAL(8,2), `nota_falta_id` INT) BEGIN     DECLARE valor_iva DECIMAL(8,2);     DECLARE iva DECIMAL(8,2);      SET @valor_iva = (valor+(valor*17)/100);     SET @iva = ((valor*17)/100);      UPDATE nota_faltas SET nota_faltas.valor_total = (nota_faltas.valor_total + valor),     nota_faltas.iva = (nota_faltas.iva + @iva),     nota_faltas.valor_iva = (nota_faltas.valor_iva + @valor_iva) WHERE nota_faltas.id = nota_falta_id; END\$\$ DELIMITER ; </pre>
3	SP_decrementar_iten_nota_faltas	<pre> DELIMITER \$\$ DROP PROCEDURE IF EXISTS `SP_decrementar_iten_nota_faltas`\$\$ CREATE PROCEDURE `SP_decrementar_iten_nota_faltas`(`produto_id` INT, `qtd` INT, `desconto` INT, `saida_id` INT) BEGIN      DECLARE qtd_rest INT;     DECLARE qtd_old INT;     DECLARE preco_venda DECIMAL(8,2);     DECLARE subtotal decimal(8,2);     DECLARE valor decimal(8,2);     DECLARE nota_falta_id INT;      SET @preco_venda = (SELECT produtos.preco_venda FROM produtos WHERE produtos.id = produto_id);     SET @nota_falta_id = (SELECT nota_faltas.id FROM nota_faltas WHERE nota_faltas.saida_id = saida_id);      SET @qtd_old = (SELECT iten_nota_faltas.quantidade FROM iten_nota_faltas </pre>

		<pre> WHERE iten_nota_faltas.nota_falta_id = @nota_falta_id AND       iten_nota_faltas.produto_id = produto_id);  SET @qtd_rest = (@qtd_old - qtd);  SET @valor = (@qtd_rest * @preco_venda); SET @subtotal = ((@qtd_rest * @preco_venda) - (@qtd_rest * @preco_venda * desconto) / 100);  UPDATE iten_nota_faltas SET       iten_nota_faltas.quantidade = @qtd_rest,       iten_nota_faltas.valor = @valor,       iten_nota_faltas.desconto = desconto,       iten_nota_faltas.subtotal = @subtotal       WHERE iten_nota_faltas.nota_falta_id = @nota_falta_id AND                                      iten_nota_faltas.produto_id = produto_id; END\$\$ DELIMITER ; </pre>
4	SP_incrementar_iten_nota_faltas	<pre> DELIMITER \$\$ DROP PROCEDURE IF EXISTS `SP_incrementar_iten_nota_faltas`\$\$ CREATE PROCEDURE `SP_incrementar_iten_nota_faltas`(`produto_id` INT, `qtd` INT, `desconto` INT, `saida_id` INT) BEGIN      DECLARE qtd_rest INT;     DECLARE qtd_old INT;     DECLARE preco_venda DECIMAL(8,2);     DECLARE subtotal decimal(8,2);     DECLARE valor decimal(8,2);     DECLARE nota_falta_id INT;      SET @preco_venda = (SELECT produtos.preco_venda FROM produtos WHERE produtos.id = produto_id);     SET @nota_falta_id = (SELECT nota_faltas.id FROM nota_faltas WHERE nota_faltas.saida_id = saida_id); </pre>

		<pre> SET @qtd_old = (SELECT iten_nota_faltas.quantidade FROM iten_nota_faltas WHERE iten_nota_faltas.nota_falta_id = @nota_falta_id AND iten_nota_faltas.produto_id = produto_id);  SET @qtd_rest = (@qtd_old + qtd);  SET @valor = (@qtd_rest * @preco_venda); SET @subtotal = ((@qtd_rest * @preco_venda) - (@qtd_rest * @preco_venda * desconto) / 100);  UPDATE iten_nota_faltas SET iten_nota_faltas.quantidade = @qtd_rest, iten_nota_faltas.valor = @valor, iten_nota_faltas.desconto = desconto, iten_nota_faltas.subtotal = @subtotal WHERE iten_nota_faltas.nota_falta_id = @nota_falta_id AND iten_nota_faltas.produto_id = produto_id; END\$\$ DELIMITER ; </pre>
5	SP_eliminar_iten_nota_faltas	<pre> DELIMITER \$\$ DROP PROCEDURE IF EXISTS `SP_eliminar_iten_nota_faltas`\$\$ CREATE PROCEDURE `SP_eliminar_iten_nota_faltas`(`produto_id` INT, `saida_id` INT) BEGIN DECLARE nota_falta_id INT;  SET @nota_falta_id = (SELECT nota_faltas.id FROM nota_faltas WHERE nota_faltas.saida_id = saida_id);  DELETE FROM iten_nota_faltas WHERE iten_nota_faltas.nota_falta_id = @nota_falta_id AND </pre>

		<pre> iten_nota_faltas.produto_id = produto_id;  END\$\$ DELIMITER ; </pre>
--	--	---

## Novos Triggers

### Triggers - Tabela "iten\_nota\_faltas"

Novos Triggers na Tabela "iten_nota_faltas"				
	Nome do Trigger	Time	Nome da Tabela	Script
1	tr_atualizar_valor_total_nota_falta_after_insert	AFTER INSERT	iten_nota_faltas	<pre> DELIMITER \$\$ DROP TRIGGER IF EXISTS `tr_atualizar_valor_total_nota_falta_after_insert`\$\$ CREATE TRIGGER `tr_atualizar_valor_total_nota_falta_after_insert` AFTER INSERT ON `iten_nota_faltas` FOR EACH ROW BEGIN     CALL SP_incrementar_valor_total_nota_falta(NEW.subtotal, NEW.nota_falta_id); END\$\$ DELIMITER ; </pre>
2	tr_atualizar_iten_nota_falta_after_update	AFTER UPDATE	iten_nota_faltas	<pre> DELIMITER \$\$ DROP TRIGGER IF EXISTS `tr_atualizar_iten_nota_falta_after_update`\$\$ CREATE TRIGGER `tr_atualizar_iten_nota_falta_after_update` AFTER UPDATE ON `iten_nota_faltas` FOR EACH ROW BEGIN      DECLARE subtotal_dif decimal;     DECLARE produto_id int;      IF NEW.subtotal &gt; OLD.subtotal THEN         SET @subtotal_dif = NEW.subtotal - OLD.subtotal;         CALL SP_incrementar_valor_total_nota_falta(@subtotal_dif, NEW.nota_falta_id);      ELSEIF NEW.subtotal &lt; OLD.subtotal THEN         SET @subtotal_dif = OLD.subtotal - NEW.subtotal; </pre>

				<pre> CALL SP_decrementar_valor_total_nota_falta(@subtotal_dif, NEW.nota_falta_id);  END IF;  END\$\$ DELIMITER ; </pre>
3	tr_atualizar_valor_total_nota_falta_after_delete	AFTER DELETE	iten_nota_faltas	<pre> DELIMITER \$\$ DROP TRIGGER IF EXISTS `tr_atualizar_valor_total_nota_falta_after_delete`\$\$ CREATE TRIGGER `tr_atualizar_valor_total_nota_falta_after_delete` AFTER DELETE ON `iten_nota_faltas` FOR EACH ROW BEGIN     CALL SP_decrementar_valor_total_nota_falta(OLD.subtotal, OLD.nota_falta_id); END\$\$ DELIMITER ; </pre>

## Atualização dos Triggers Existentes

### Triggers – Tabela “iten\_saidas”

Atualização dos Triggers Existentes, Triggers na Tabela “iten_saidas”				
	Trigger	Time	Tabela	Script
1	tr_atualizar_valor_total_saida_after_insert	AFTER INSERT	iten_saidas	<pre> DELIMITER \$\$ DROP TRIGGER IF EXISTS `tr_atualizar_valor_total_saida_after_insert`\$\$ CREATE TRIGGER `tr_atualizar_valor_total_saida_after_insert` AFTER INSERT ON `iten_saidas` FOR EACH ROW BEGIN      CALL SP_incrementar_valor_total_saida(NEW.subtotal, NEW.saida_id);     CALL SP_incrementar_iten_nota_faltas(NEW.produto_id, NEW.quantidade, NEW.desconto, NEW.saida_id); END DELIMITER ; </pre>
2	tr_atualizar_iten_saidas_after_update	AFTER UPDATE	iten_saidas	<pre> DELIMITER \$\$ DROP TRIGGER IF EXISTS `tr_atualizar_iten_saidas_after_update`\$\$ </pre>

				<pre> CREATE TRIGGER `tr_atualizar_iten_saidas_after_update` AFTER UPDATE ON `iten_saidas` FOR EACH ROW BEGIN      DECLARE quantidade INT;      IF NEW.quantidade &gt; OLD.quantidade THEN         SET @quantidade = NEW.quantidade - OLD.quantidade;         CALL SP_incrementar_iten_nota_faltas(NEW.produto_id, @quantidade, NEW.desconto, NEW.saida_id);      ELSEIF NEW.quantidade &lt; OLD.quantidade THEN         SET @quantidade = OLD.quantidade - NEW.quantidade;         CALL SP_decrementar_iten_nota_faltas(NEW.produto_id, @quantidade, NEW.desconto, NEW.saida_id);     END IF;     /* Nao temos igual porque garante que o iten_nota_faltas possa ser actualizado correctamente, pois haveria conflito a partir da actualizacao do iten_guiarentrega que por sua vez chama o actualizar_rest_iten_saidas*/      IF NEW.subtotal &gt; OLD.subtotal THEN         SET @subtotal_dif = NEW.subtotal - OLD.subtotal;         CALL SP_incrementar_valor_total_saida(@subtotal_dif, NEW.saida_id);      ELSEIF NEW.subtotal &lt; OLD.subtotal THEN         SET @subtotal_dif = OLD.subtotal - NEW.subtotal;         CALL SP_decrementar_valor_total_saida(@subtotal_dif, NEW.saida_id);      END IF; END\$\$ DELIMITER ; </pre>
3	tr_atualizar_valor_total_saida_after_delete	AFTER DELETE	iten_saidas	<pre> DELIMITER \$\$ DROP TRIGGER IF EXISTS `tr_atualizar_valor_total_saida_after_delete`\$\$ CREATE TRIGGER `tr_atualizar_valor_total_saida_after_delete` AFTER DELETE ON `iten_saidas` FOR EACH ROW BEGIN     CALL SP_decrementar_valor_total_saida(OLD.subtotal, OLD.saida_id); </pre>





2	tr_atualizar_re st_iten_saidas_af ter_update	AFTER UPDATE	iten_guiaent regas	<pre> DELIMITER \$\$ DROP TRIGGER IF EXISTS `tr_atualizar_rest_iten_saidas_after_update`\$\$ CREATE TRIGGER `tr_atualizar_rest_iten_saidas_after_update` AFTER UPDATE ON `iten_guiaentregas` FOR EACH ROW BEGIN     DECLARE saida_id int;     DECLARE produto_id int;     DECLARE quantidade int;     DECLARE subtotal decimal;      SET @saida_id = (SELECT guia_entregas.saida_id FROM guia_entregas, iten_guiaentregas                                 WHERE iten_guiaentregas.guia_entrega_id = guia_entregas.id AND                                 iten_guiaentregas.guia_entrega_id = NEW.guia_entrega_id AND                                 iten_guiaentregas.produto_id = NEW.produto_id);      SET @produto_id = NEW.produto_id;      IF NEW.quantidade &gt; OLD.quantidade THEN         SET @quantidade = NEW.quantidade - OLD.quantidade;         CALL SP_decrementar_qtd_produto(@quantidade, NEW.produto_id);         CALL SP_decrementar_rest_iten_saidas(@quantidade, @saida_id, @produto_id);         CALL SP_decrementar_iten_nota_faltas(@produto_id, @quantidade, NEW.desconto, @saida_id);      ELSEIF NEW.quantidade &lt; OLD.quantidade THEN         SET @quantidade = OLD.quantidade - NEW.quantidade;         CALL SP_incrementar_qtd_produto(@quantidade, NEW.produto_id);         CALL SP_incrementar_rest_iten_saidas(@quantidade, @saida_id, @produto_id);         CALL SP_incrementar_iten_nota_faltas(@produto_id, @quantidade, NEW.desconto, @saida_id);     END IF; </pre>
---	--	-----------------	-----------------------	--

				<pre> IF NEW.subtotal &gt; OLD.subtotal THEN     SET @subtotal = NEW.subtotal - OLD.subtotal;     CALL SP_incrementar_valor_total_guia_entrega(@subtotal, NEW.guia_entrega_id);  ELSEIF NEW.subtotal &lt; OLD.subtotal THEN     SET @subtotal = OLD.subtotal - NEW.subtotal;     CALL SP_decrementar_valor_total_guia_entrega(@subtotal, NEW.guia_entrega_id); END IF;  END\$\$ DELIMITER ; </pre>
3	tr_actuatlizar_re st_iten_saidas_be fore_delete	BEFORE DELETE	iten_guiaent regas	<pre> DELIMITER \$\$ DROP TRIGGER IF EXISTS `tr_actuatlizar_rest_iten_saidas_before_delete`\$\$ CREATE TRIGGER `tr_actuatlizar_rest_iten_saidas_before_delete` BEFORE DELETE ON `iten_guiaentregas` FOR EACH ROW BEGIN     DECLARE saida_id int;     DECLARE produto_id int;     DECLARE nota_falta_id int;      SET @saida_id = (SELECT DISTINCT guia_entregas.saida_id FROM guia_entregas, iten_guiaentregas                         WHERE guia_entregas.id = OLD.guia_entrega_id AND                         iten_guiaentregas.produto_id = OLD.produto_id);      SET @nota_falta_id = (SELECT nota_faltas.id FROM nota_faltas WHERE nota_faltas.saida_id = @saida_id);      SET @produto_id = OLD.produto_id;      CALL SP_incrementar_rest_iten_saidas(OLD.quantidade, @saida_id, @produto_id);     CALL SP_incrementar_iten_nota_faltas(@produto_id, OLD.quantidade, OLD.desconto, @saida_id); </pre>

				CALL SP_incrementar_qtd_produto(OLD.quantidade, @produto_id); END\$\$ DELIMITER ;
--	--	--	--	---

#### OUTROS ACRESCIMOS/ACTUALIZAÇÕES:

É necessário inserir a permissão “visualizar\_nota\_de\_falta” na tabela permissões pelo SGBD e atribuir ao tipo de usuário esta permissão no Sistema para que o mesmo possa aceder as notas de falta pelo Sistema;