Projeto de Banco de Dados - Eficiência na Armazenagem Logística

Sumário

---

1. Introdução

O Projeto de Armazenagem Eficiente busca otimizar a gestão de armazéns através da implementação de um sistema focado em: melhor aproveitamento do espaço, controle de estoque aprimorado e maior eficiência nas operações de recebimento, picking, packing e expedição.

---

2. Objetivos

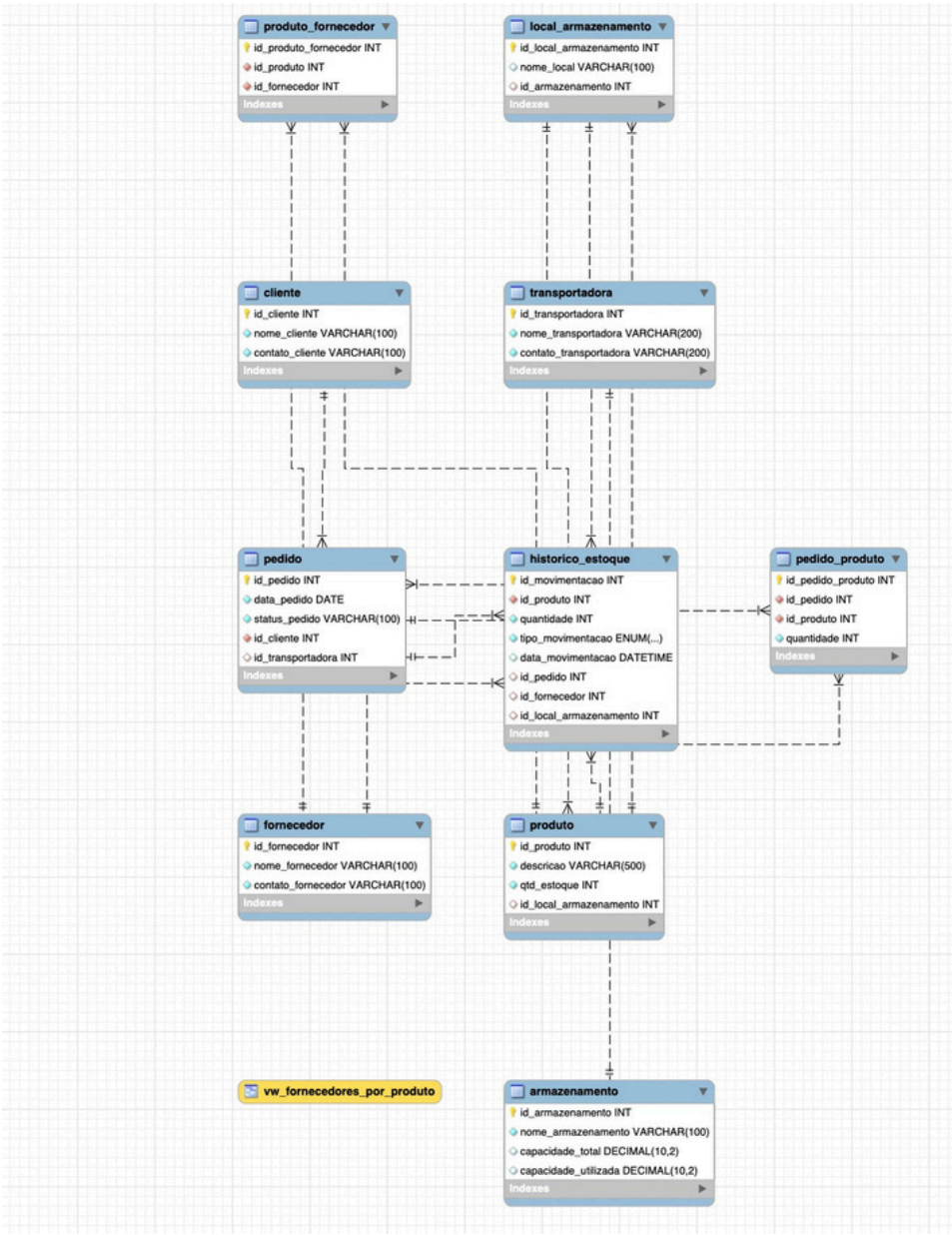
- Otimização do espaço de armazenamento.
- Aprimoramento da precisão e da eficiência nas operações de separação e embalagem.
- Agilização dos procedimentos de expedição e recebimento.
- Facilitação da gestão de inventário e da rastreabilidade de mercadorias.

---

3. Estrutura do Banco de Dados

- Banco: armazen
- Tabelas:
  - Armazenamento
  - Produto
  - Pedido
  - Transportadora
  - Fornecedor

3.3 Diagrama de Entidade-Relacionamento



```
drop database if exists armazem;
create database armazem;
use armazem;

-- Tabelas básicas
create table armazenamento (
    id_armazenamento int not null auto_increment primary key,
    nome_armazenamento varchar(100) not null,
    capacidade_total decimal(10,2),
    capacidade_utilizada decimal(10,2)
);

create table local_armazenamento (
    id_local_armazenamento int auto_increment primary key,
    nome_local varchar(100),
    id_armazenamento int,
    foreign key (id_armazenamento) references armazenamento(id_armazenamento)
);

create table cliente (
    id_cliente int auto_increment primary key,
    nome_cliente varchar(100) not null,
    contato_cliente varchar(100) not null
);

create table produto (
```

```
    id_produto int auto_increment primary key,
    descricao varchar(500) not null,
    qtd_estoque int not null,
    id_local_armazenamento int,
    foreign key (id_local_armazenamento) references
local_armazenamento(id_local_armazenamento)
);

create table transportadora (
    id_transportadora int auto_increment primary key,
    nome_transportadora varchar(200) not null,
    contato_transportadora varchar(200) not null
);

create table fornecedor (
    id_fornecedor int auto_increment primary key,
    nome_fornecedor varchar(100) not null,
    contato_fornecedor varchar(100) not null
);

create table pedido (
    id_pedido int auto_increment primary key,
    data_pedido date not null,
    status_pedido varchar(100) not null,
    id_cliente int not null,
    id_transportadora int,
    foreign key (id_cliente) references cliente(id_cliente),
```

```
    foreign key (id_transportadora) references transportadora(id_transportadora)
);

-- Tabela de relacionamento entre pedido e produto
create table pedido_produto (
    id_pedido_produto int auto_increment primary key,
    id_pedido int not null,
    id_produto int not null,
    quantidade int not null,
    foreign key (id_pedido) references pedido(id_pedido),
    foreign key (id_produto) references produto(id_produto)
);

-- Tabela de relacionamento entre produto e fornecedor
create table produto_fornecedor (
    id_produto_fornecedor int auto_increment primary key,
    id_produto int not null,
    id_fornecedor int not null,
    foreign key (id_produto) references produto(id_produto),
    foreign key (id_fornecedor) references fornecedor(id_fornecedor)
);

-- Inserção de dados
insert into armazenamento
(nome_armazenamento, capacidade_total, capacidade_utilizada)
values
("Armazém Central","5000.00","3200.00"),
```

("Depósito Norte","2500.00","1800.00"),

("CD Logística SP","8000.00","6500.00"),

("Estoque Rápido","1200.00","900.00"),

("Armazém Portuário","10000.00","4200.00"),

("Centro Dist. RJ","6000","6000"),

("Cold Storage","1500","1200"),

("Depósito Seco","3000.00","1500.00"),

("Artazém Automatizado","7500.00","5000.00"),

("Estoque Temporário","800.00","800.00");

insert into local_armazenamento

(nome_local, id_armazenamento)

values

("Galpão 1", 1),

("Galpão 2", 1),

("Galpão 3", 2),

("Galpão 4", 3),

("Galpão 5", 4),

("Galpão 6", 5),

("Galpão 7", 6),

("Galpão 8", 7),

("Galpão 9", 8),

("Galpão 10", 9);

insert into cliente

(id_cliente, nome_cliente, contato_cliente)

values

(1,"Indústria ABC", "contato@industriaabc.com.br"),

(2,"Metalúrgica XYZ", "vendas@metalurgicaxyz.com.br"),

(3,"Automação Total", "sac@automatotal.com.br"),

(4,"RoboTech Solutions", "comercial@robotech.com.br"),

(5,"PlasticForm Indústria", "contato@plasticform.com.br"),

(6,"EletroMecânica Brasil", "vendas@eletromecanica.com.br"),

(7,"AutoParts Ltda", "sac@autoparts.com.br"),

(8,"Mecânica Pesada SA", "comercial@mecanicapesada.com.br"),

(9,"TecnoIndústria", "contato@tecnoindustria.com.br"),

(10,"Precision Tools", "vendas@precisiontools.com.br");

insert into produto

(descricao, qtd_estoque, id_local_armazenamento)

values

("Controlador Lógico Programável - CLP modular com entradas/saídas digitais e analógicas, comunicação PROFINET","200", 1),

("Sensor de Proximidade Indutivo - Sensor de detecção de metais sem contato, alcance de 5mm, saída PNP","390", 2),

("Inversor de Freqência Weg - Acionamento de motores elétricos com controle de velocidade e torque","50", 3),

("HMI - Tela touchscreen de 7'' para supervisão e controle de processos","500", 4),

("Servomotor Schneider Electric Lexium - Motor de alto desempenho para posicionamento preciso em robótica","377", 5),

("Barramento de Campo - Módulo para integração de dispositivos em redes industriais","800", 6),

("Controlador de Temperatura Omron - Controlador PID digital para automação de processos térmicos","88", 7),

("Robô Industrial - Robô articulado para paletização e manuseio de cargas","150", 8),

("Rede de Sensores Sem Fio - (IoT) - Kit para monitoramento remoto de variáveis industriais (pressão, temperatura, vibração)","40", 9),

("Software de Supervisão - Plataforma para visualizasão, controle e análise de dados em automação","90", 10);

INSERT INTO pedido

(id_cliente,data_pedido, status_pedido)

VALUES

(1,'2024-08-01','Pendente'),

(2,'2024-03-30','Em processamento'),

(3,'2023-01-10','Pagamento aprovado'),

(4,'2022-07-20','Entregue'),

(5,'2024-10-09','Enviado'),

(6,'2025-02-09','Pagamento aprovado'),

(7,'2022-10-10','Cancelado'),

(8,'2023-08-25','Enviado'),

(9,'2024-05-18','Pendente'),

(10,'2023-04-29','Em processamento');

insert into fornecedor

(id_fornecedor, nome_fornecedor, contato_fornecedor)

values

("1","RoboCore Solutions","vendas@robocore.automation"),

("2","AutoLink Tecnologia","cotacao@autolink.tech"),

("3","PlcFactory Integração","projetos@plcfactory.com"),

("4","IoT Industrial Labs","suporte@iotlabs.automation"),

("5","EletroConex Distribuidora","pedidos@eletroconex.com.br"),

("6","Pneumatech Components","comercial@pneumatech.eng.br"),

("7","CodeFactory Automação","licensing@codefactory.io"),

("8","NeuralAutomate AI","contato@neuralautomate.tech"),

("9","ManuTech Consultoria","consultoria@manutech.eng"),

("10","AutoMan Manutenção","os@automan.maintenance");

insert into produto_fornecedor

(id_produto, id_fornecedor)

values

(1, 1),

(1, 2),

(2, 3),

(3, 4),

(4, 5),

(5, 6),

(6, 7),

(7, 8),

(8, 9),

(9, 10),

(10, 1),

(2, 4),

(3, 5);

INSERT INTO

pedido_produto (id_pedido, id_produto, quantidade)

VALUES

(1, 1, 5),

(1, 2, 10),

(2, 3, 2),

```
(3, 4, 1),
(4, 5, 3),
(5, 6, 8),
(6, 7, 4),
(7, 8, 1),
(8, 9, 2),
(9, 10, 1);

insert into transportadora
(id_transportadora,nome_transportadora, contato_transportadora)
values
(1,"Rápido Express Logística","atendimento@rapidoexpress.com.br"),
(2,"Total Cargas Transportes","sac@totalcargas.com.br"),
(3,"Brasil Norte Entregas","contato@brasilnorte.com.br"),
(4,"Veloz Transportadora Nacional","suporte@veloztransporte.com.br"),
(5,"Logística Sul Americana","faleconosco@logisticasul.com.br"),
(6,"Cargo Seguro Transportes","comercial@cargoseguro.com.br"),
(7,"Expresso Nacional","atendimento@expressonacional.com.br"),
(8,"TransOeste Logística","ouvidoria@transoeste.com.br"),
(9,"Global Delivery Express","helpdesk@globaldelivery.com.br"),
(10,"Azul Cargas Rápidas","sac@azulcargas.com.br");

-- Associação entre produtos e seus locais de armazenamento.
SELECT
    p.id_produto,
    p.descricao AS produto,
    la.nome_local AS local_armazenamento,
```

```
    a.nome_armazenamento,
    a.capacidade_total,
    a.capacidade_utilizada
FROM
    produto p
JOIN
    local_armazenamento la ON p.id_local_armazenamento = la.id_local_armazenamento
JOIN
    armazenamento a ON la.id_armazenamento = a.id_armazenamento;

-- Vínculo entre pedidos, produtos e clientes.
SELECT
    pp.id_pedido,
    c.nome_cliente,
    pr.descricao AS produto,
    pp.quantidade,
    pe.data_pedido
FROM
    pedido_produto pp
LEFT JOIN pedido pe ON pp.id_pedido = pe.id_pedido
LEFT JOIN cliente c ON pe.id_cliente = c.id_cliente
LEFT JOIN produto pr ON pp.id_produto = pr.id_produto
ORDER BY pe.data_pedido DESC;

-- Registro de fornecedores associados a produtos específicos.
SELECT
    p.id_produto,
```

```
    p.descricao AS produto,
    f.id_fornecedor,
    f.nome_fornecedor,
    f.contato_fornecedor
FROM
    produto_fornecedor pf
JOIN
    produto p ON pf.id_produto = p.id_produto
JOIN
    fornecedor f ON pf.id_fornecedor = f.id_fornecedor
ORDER BY
    p.id_produto;


    -- 4.1. Controle de Estoque
-- Criação da tabela de histórico de estoque
CREATE TABLE IF NOT EXISTS historico_estoque (
    id_movimentacao INT AUTO_INCREMENT PRIMARY KEY,
    id_produto INT NOT NULL,
    quantidade INT NOT NULL,
    tipo_movimentacao ENUM('Entrada', 'Saída') NOT NULL,
    data_movimentacao DATETIME DEFAULT CURRENT_TIMESTAMP,
    id_pedido INT,
    id_fornecedor INT,
    id_local_armazenamento INT,
    FOREIGN KEY (id_produto) REFERENCES produto(id_produto),
    FOREIGN KEY (id_pedido) REFERENCES pedido(id_pedido),
    FOREIGN KEY (id_fornecedor) REFERENCES fornecedor(id_fornecedor),
```

```
    FOREIGN KEY (id_local_armazenamento) REFERENCES
local_armazenamento(id_local_armazenamento)

);


-- Procedure para atualizar o estoque

DELIMITER //

CREATE PROCEDURE AtualizarEstoque(

    IN p_id_produto INT,

    IN p_quantidade INT,

    IN p_tipo_movimentacao VARCHAR(10),

    IN p_id_pedido INT,

    IN p_id_fornecedor INT,

    IN p_id_local_armazenamento INT

)

BEGIN

    DECLARE v_quantidade_atual INT;

    DECLARE v_capacidade_disponivel DECIMAL(10,2);


    IF p_tipo_movimentacao = 'Entrada' THEN

        UPDATE produto

        SET qtd_estoque = qtd_estoque + p_quantidade

        WHERE id_produto = p_id_produto;


        UPDATE armazenamento a

        JOIN local_armazenamento la ON a.id_armazenamento = la.id_armazenamento

        SET a.capacidade_utilizada = a.capacidade_utilizada + p_quantidade

        WHERE la.id_local_armazenamento = p_id_local_armazenamento;
```

```
    ELSEIF p_tipo_movimentacao = 'Saída' THEN
        SELECT qtd_estoque INTO v_quantidade_atual
        FROM produto
        WHERE id_produto = p_id_produto;

        IF v_quantidade_atual < p_quantidade THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'ERRO: Estoque insuficiente!';
        END IF;

        UPDATE produto
        SET qtd_estoque = qtd_estoque - p_quantidade
        WHERE id_produto = p_id_produto;
    END IF;

    INSERT INTO historico_estoque (
        id_produto, quantidade, tipo_movimentacao,
        id_pedido, id_fornecedor, id_local_armazenamento
    ) VALUES (
        p_id_produto, p_quantidade, p_tipo_movimentacao,
        p_id_pedido, p_id_fornecedor, p_id_local_armazenamento
    );

    IF (SELECT qtd_estoque FROM produto WHERE id_produto = p_id_produto) < 5 THEN
        SELECT CONCAT('ALERTA: Estoque do produto #', p_id_produto, ' está abaixo do mínimo!') AS
aviso;
```

```
    END IF;
END //
DELIMITER ;

-- Procedure para alocar produto automaticamente
DELIMITER //
CREATE PROCEDURE AlocarLocalArmazenamento(
    IN p_id_produto INT,
    IN p_quantidade DECIMAL(10,2)
)
BEGIN
    DECLARE v_id_local INT;

    SELECT la.id_local_armazenamento INTO v_id_local
    FROM local_armazenamento la
    JOIN armazenamento a ON la.id_armazenamento = a.id_armazenamento
    WHERE (a.capacidade_total - a.capacidade_utilizada) >= p_quantidade
    ORDER BY (a.capacidade_total - a.capacidade_utilizada) ASC
    LIMIT 1;

    IF v_id_local IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'ERRO: Nenhum local com espaço suficiente!';
    ELSE
        UPDATE produto
        SET id_local_armazenamento = v_id_local
        WHERE id_produto = p_id_produto;
```

```
        SELECT CONCAT('Produto alocado no Local #', v_id_local) AS resultado;
    END IF;
END //
DELIMITER ;

-- Procedure para processar pedidos
DELIMITER //
CREATE PROCEDURE ProcessarPedido(IN p_id_pedido INT)
BEGIN
    DECLARE v_id_produto INT;
    DECLARE v_quantidade INT;
    DECLARE v_id_local INT;
    DECLARE done INT DEFAULT FALSE;

    DECLARE cur_itens CURSOR FOR
        SELECT id_produto, quantidade
        FROM pedido_produto
        WHERE id_pedido = p_id_pedido;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur_itens;

    itens_loop: LOOP
        FETCH cur_itens INTO v_id_produto, v_quantidade;
        IF done THEN
```

```
            LEAVE itens_loop;
        END IF;

        SELECT id_local_armazenamento INTO v_id_local
        FROM produto
        WHERE id_produto = v_id_produto;

        CALL AtualizarEstoque(v_id_produto, v_quantidade, 'Saída', p_id_pedido, NULL, v_id_local);
    END LOOP;

    CLOSE cur_itens;

    UPDATE pedido SET status_pedido = 'Processado' WHERE id_pedido = p_id_pedido;

    SELECT CONCAT('Pedido #', p_id_pedido, ' processado com sucesso!') AS resultado;
END //
DELIMITER ;

-- View de fornecedores por produto
CREATE VIEW vw_fornecedores_por_produto AS
SELECT
    p.id_produto,
    p.descricao AS produto,
    f.id_fornecedor,
    f.nome_fornecedor,
    f.contato_fornecedor
FROM produto p
```

```
JOIN produto_fornecedor pf ON p.id_produto = pf.id_produto

JOIN fornecedor f ON pf.id_fornecedor = f.id_fornecedor

ORDER BY p.id_produto;


-- Inserir entrada de estoque

CALL AtualizarEstoque(1, 50, 'Entrada', NULL, 1, 1);


-- Inserir saída de estoque vinculada a pedido

CALL AtualizarEstoque(1, 10, 'Saída', 1, NULL, 1);


-- Verificar estoque atualizado

SELECT * FROM produto WHERE id_produto = 1;


-- Verificar histórico completo

SELECT * FROM historico_estoque;


-- Alocar produto dinamicamente

CALL AlocarLocalArmazenamento(2, 100);


-- Verificar local atribuído

SELECT p.id_produto, p.descricao, la.nome_local

FROM produto p

JOIN local_armazenamento la ON p.id_local_armazenamento = la.id_local_armazenamento

WHERE p.id_produto = 6;


-- Processar pedido completo

CALL ProcessarPedido(10);
```

```
-- Verificar pedido após processamento

SELECT * FROM pedido WHERE id_pedido = 1;


-- Verificar movimentações daquele pedido

SELECT * FROM historico_estoque WHERE id_pedido = 1;


-- Consultar fornecedores vinculados a um produto

SELECT * FROM vw_fornecedores_por_produto WHERE id_produto = 1;
```

6. Conclusão

O Projeto de Armazenagem Eficiente propõe uma solução abrangente para melhorar a eficiência e controle nas operações logísticas. A estrutura do banco de dados é projetada para ser flexível, segura e capaz de suportar o crescimento futuro da organização.