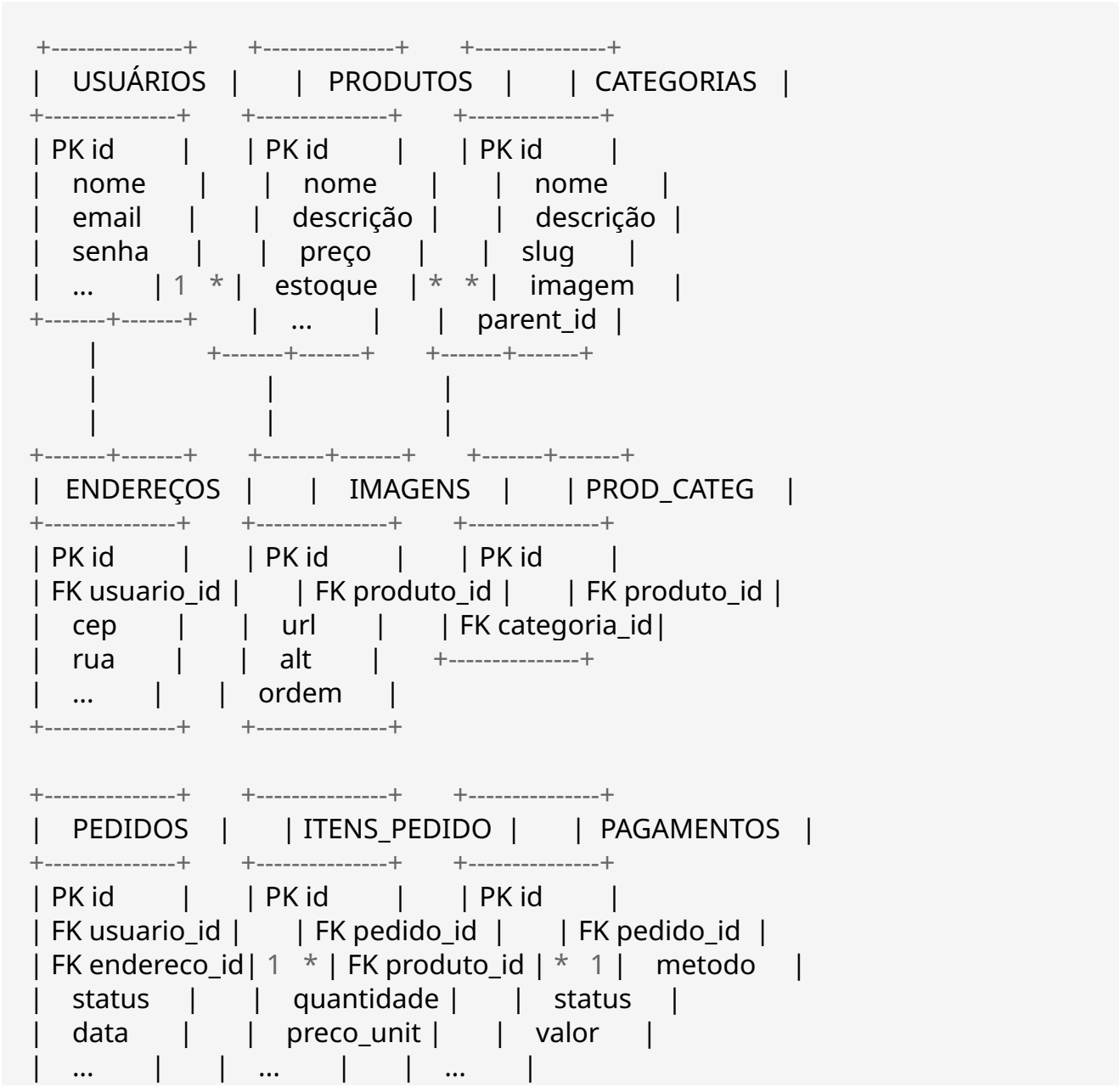
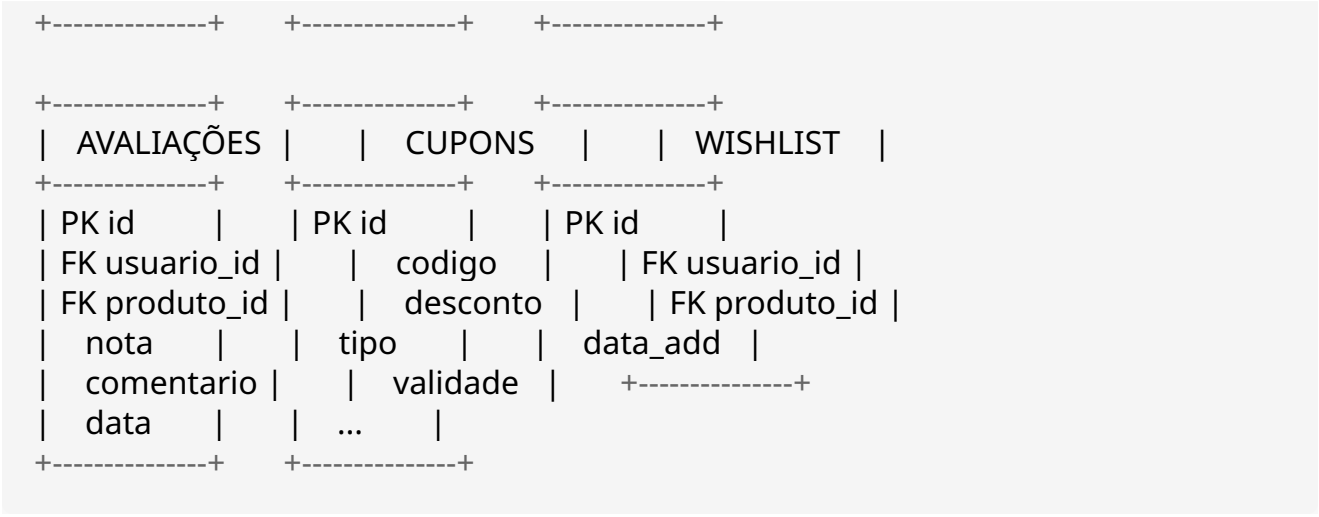
Estrutura do Banco de Dados MySQL para E-commerce

Visão Geral

Este documento detalha o planejamento da estrutura do banco de dados MySQL para suportar todas as funcionalidades do e-commerce. O design segue os princípios de normalização até a 3ª forma normal, garantindo integridade dos dados e minimizando redundâncias, enquanto mantém a performance para consultas frequentes.

Diagrama de Entidade-Relacionamento (Conceitual)





Detalhamento das Tabelas

1. Usuários

Armazena informações dos clientes e administradores do sistema.

```
CREATE TABLE usuarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE,
  senha VARCHAR(255) NOT NULL,
  cpf VARCHAR(14) UNIQUE,
  telefone VARCHAR(20),
  data_nascimento DATE,
  tipo ENUM('cliente', 'admin') DEFAULT 'cliente',
  status ENUM('ativo', 'inativo', 'bloqueado') DEFAULT 'ativo',
  data_cadastro DATETIME DEFAULT CURRENT_TIMESTAMP,
  data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  ultimo_login DATETIME,
  token_reset_senha VARCHAR(100),
  expiracao_token DATETIME,
  preferencias JSON,
  INDEX idx_email (email),
  INDEX idx_tipo (tipo)
);
```

2. Endereços

Armazena os endereços de entrega e cobrança dos usuários.

```
CREATE TABLE enderecos (
  id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    usuario_id INT NOT NULL,
    nome VARCHAR(100) NOT NULL,
    cep VARCHAR(9) NOT NULL,
    logradouro VARCHAR(100) NOT NULL,
    numero VARCHAR(20) NOT NULL,
    complemento VARCHAR(100),
    bairro VARCHAR(100) NOT NULL,
    cidade VARCHAR(100) NOT NULL,
    estado CHAR(2) NOT NULL,
    pais VARCHAR(50) DEFAULT 'Brasil',
    telefone VARCHAR(20),
    tipo ENUM('entrega', 'cobranca', 'ambos') DEFAULT 'entrega',
    padrao BOOLEAN DEFAULT FALSE,
    data_cadastro DATETIME DEFAULT CURRENT_TIMESTAMP,
    data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE,
    INDEX idx_usuario_id (usuario_id)
);
```

3. Categorias

Armazena as categorias de produtos, permitindo estrutura hierárquica.

```
 CREATE TABLE categorias (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    descricao TEXT,
    slug VARCHAR(100) NOT NULL UNIQUE,
    imagem VARCHAR(255),
    banner VARCHAR(255),
    parent_id INT,
    nivel INT DEFAULT 1,
    ordem INT DEFAULT 0,
    status ENUM('ativo', 'inativo') DEFAULT 'ativo',
    meta_titulo VARCHAR(100),
    meta_descricao VARCHAR(255),
    data_cadastro DATETIME DEFAULT CURRENT_TIMESTAMP,
    data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (parent_id) REFERENCES categorias(id) ON DELETE SET NULL,
    INDEX idx_slug (slug),
    INDEX idx_parent_id (parent_id),
    INDEX idx_status (status)
);
```

4. Produtos

Armazena informações detalhadas sobre os produtos disponíveis.

```
CREATE TABLE produtos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  sku VARCHAR(50) NOT NULL UNIQUE,
  nome VARCHAR(255) NOT NULL,
  descricao_curta VARCHAR(255),
  descricao_longa TEXT,
  preco DECIMAL(10, 2) NOT NULL,
  preco_promocional DECIMAL(10, 2),
  estoque INT NOT NULL DEFAULT 0,
  peso DECIMAL(10, 2),
  dimensoes_altura DECIMAL(10, 2),
  dimensoes_largura DECIMAL(10, 2),
  dimensoes_profundidade DECIMAL(10, 2),
  destaque BOOLEAN DEFAULT FALSE,
  novo BOOLEAN DEFAULT FALSE,
  status ENUM('ativo', 'inativo', 'esgotado') DEFAULT 'ativo',
  gerenciar_estoque BOOLEAN DEFAULT TRUE,
  permitir_sem_estoque BOOLEAN DEFAULT FALSE,
  slug VARCHAR(255) NOT NULL UNIQUE,
  meta_titulo VARCHAR(100),
  meta_descricao VARCHAR(255),
  data_cadastro DATETIME DEFAULT CURRENT_TIMESTAMP,
  data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  visualizacoes INT DEFAULT 0,
  vendas INT DEFAULT 0,
  atributos JSON,
  INDEX idx_sku (sku),
  INDEX idx_nome (nome),
  INDEX idx_slug (slug),
  INDEX idx_status (status),
  INDEX idx_destaque (destaque),
  INDEX idx_preco (preco)
);
```

5. Produto_Categorias

Tabela de relacionamento muitos-para-muitos entre produtos e categorias.

```
CREATE TABLE produto_categorias (
  id INT AUTO_INCREMENT PRIMARY KEY,
  produto_id INT NOT NULL,
  categoria_id INT NOT NULL,
  principal BOOLEAN DEFAULT FALSE,
  FOREIGN KEY (produto_id) REFERENCES produtos(id) ON DELETE CASCADE,
```

```
    FOREIGN KEY (categoria_id) REFERENCES categorias(id) ON DELETE CASCADE,
    UNIQUE KEY unique_produto_categoria (produto_id, categoria_id),
    INDEX idx_produto_id (produto_id),
    INDEX idx_categoria_id (categoria_id)
);
```

6. Imagens_Produto

Armazena múltiplas imagens para cada produto.

```
CREATE TABLE imagens_produto (
    id INT AUTO_INCREMENT PRIMARY KEY,
    produto_id INT NOT NULL,
    url VARCHAR(255) NOT NULL,
    alt VARCHAR(100),
    titulo VARCHAR(100),
    ordem INT DEFAULT 0,
    principal BOOLEAN DEFAULT FALSE,
    data_cadastro DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (produto_id) REFERENCES produtos(id) ON DELETE CASCADE,
    INDEX idx_produto_id (produto_id),
    INDEX idx_principal (principal)
);
```

7. Atributos

Armazena os tipos de atributos disponíveis para produtos (cor, tamanho, etc).

```
CREATE TABLE atributos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(50) NOT NULL,
    tipo ENUM('texto', 'numero', 'boolean', 'selecao') NOT NULL,
    descricao VARCHAR(255),
    data_cadastro DATETIME DEFAULT CURRENT_TIMESTAMP,
    data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    INDEX idx_nome (nome)
);
```

8. Valores_Atributo

Armazena os valores possíveis para atributos do tipo seleção.

```
CREATE TABLE valores_atributo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    atributo_id INT NOT NULL,
```

```
    valor VARCHAR(100) NOT NULL,
    ordem INT DEFAULT 0,
    FOREIGN KEY (atributo_id) REFERENCES atributos(id) ON DELETE CASCADE,
    INDEX idx_atributo_id (atributo_id)
);
```

9. Produto_Atributos

Relaciona produtos com seus atributos específicos.

```
CREATE TABLE produto_atributos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    produto_id INT NOT NULL,
    atributo_id INT NOT NULL,
    valor_id INT,
    valor_texto VARCHAR(255),
    valor_numero DECIMAL(10, 2),
    valor_boolean BOOLEAN,
    FOREIGN KEY (produto_id) REFERENCES produtos(id) ON DELETE CASCADE,
    FOREIGN KEY (atributo_id) REFERENCES atributos(id) ON DELETE CASCADE,
    FOREIGN KEY (valor_id) REFERENCES valores_atributo(id) ON DELETE SET NULL,
    INDEX idx_produto_id (produto_id),
    INDEX idx_atributo_id (atributo_id)
);
```

10. Variantes_Produto

Armazena variações de um produto (diferentes cores, tamanhos, etc).

```
CREATE TABLE variantes_produto (
    id INT AUTO_INCREMENT PRIMARY KEY,
    produto_id INT NOT NULL,
    sku VARCHAR(50) NOT NULL UNIQUE,
    preco DECIMAL(10, 2),
    preco_promocional DECIMAL(10, 2),
    estoque INT NOT NULL DEFAULT 0,
    imagem VARCHAR(255),
    status ENUM('ativo', 'inativo') DEFAULT 'ativo',
    data_cadastro DATETIME DEFAULT CURRENT_TIMESTAMP,
    data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (produto_id) REFERENCES produtos(id) ON DELETE CASCADE,
    INDEX idx_produto_id (produto_id),
    INDEX idx_sku (sku)
);
```

11. Variante_Atributos

Relaciona variantes com seus atributos específicos.

```
CREATE TABLE variante_atributos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  variante_id INT NOT NULL,
  atributo_id INT NOT NULL,
  valor_id INT,
  FOREIGN KEY (variante_id) REFERENCES variantes_produto(id) ON DELETE
CASCADE,
  FOREIGN KEY (atributo_id) REFERENCES atributos(id) ON DELETE CASCADE,
  FOREIGN KEY (valor_id) REFERENCES valores_atributo(id) ON DELETE CASCADE,
  INDEX idx_variante_id (variante_id),
  INDEX idx_atributo_id (atributo_id)
);
```

12. Pedidos

Armazena informações sobre os pedidos realizados.

```
CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  codigo VARCHAR(20) NOT NULL UNIQUE,
  usuario_id INT,
  endereco_entrega_id INT,
  endereco_cobranca_id INT,
  status ENUM('aguardando_pagamento', 'pagamento_aprovado',
'em_preparacao', 'enviado', 'entregue', 'cancelado', 'devolvido') DEFAULT
'aguardando_pagamento',
  subtotal DECIMAL(10, 2) NOT NULL,
  desconto DECIMAL(10, 2) DEFAULT 0,
  frete DECIMAL(10, 2) DEFAULT 0,
  total DECIMAL(10, 2) NOT NULL,
  metodo_envio VARCHAR(50),
  codigo_rastreamento VARCHAR(50),
  notas TEXT,
  notas_admin TEXT,
  ip_comprador VARCHAR(45),
  user_agent TEXT,
  data_pedido DATETIME DEFAULT CURRENT_TIMESTAMP,
  data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  data_pagamento DATETIME,
  data_envio DATETIME,
  data_entrega DATETIME,
  data_cancelamento DATETIME,
  cupom_id INT,
  FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE SET NULL,
```

```
    FOREIGN KEY (endereco_entrega_id) REFERENCES enderecos(id) ON DELETE
SET NULL,
    FOREIGN KEY (endereco_cobranca_id) REFERENCES enderecos(id) ON DELETE
SET NULL,
    INDEX idx_codigo (codigo),
    INDEX idx_usuario_id (usuario_id),
    INDEX idx_status (status),
    INDEX idx_data_pedido (data_pedido)
);
```

13. Itens_Pedido

Armazena os itens individuais de cada pedido.

```
 CREATE TABLE itens_pedido (
    id INT AUTO_INCREMENT PRIMARY KEY,
    pedido_id INT NOT NULL,
    produto_id INT,
    variante_id INT,
    nome_produto VARCHAR(255) NOT NULL,
    sku VARCHAR(50) NOT NULL,
    quantidade INT NOT NULL,
    preco_unitario DECIMAL(10, 2) NOT NULL,
    subtotal DECIMAL(10, 2) NOT NULL,
    atributos JSON,
    FOREIGN KEY (pedido_id) REFERENCES pedidos(id) ON DELETE CASCADE,
    FOREIGN KEY (produto_id) REFERENCES produtos(id) ON DELETE SET NULL,
    FOREIGN KEY (variante_id) REFERENCES variantes_produto(id) ON DELETE SET
NULL,
    INDEX idx_pedido_id (pedido_id),
    INDEX idx_produto_id (produto_id)
);
```

14. Pagamentos

Armazena informações sobre os pagamentos dos pedidos.

```
 CREATE TABLE pagamentos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    pedido_id INT NOT NULL,
    transacao_id VARCHAR(100),
    metodo ENUM('cartao_credito', 'boleto', 'pix', 'transferencia', 'mercadopago')
NOT NULL,
    status ENUM('pendente', 'aprovado', 'recusado', 'estornado', 'em_analise')
DEFAULT 'pendente',
    valor DECIMAL(10, 2) NOT NULL,
    parcelas INT DEFAULT 1,
    gateway VARCHAR(50) DEFAULT 'mercadopago',
```

```
    dados_pagamento JSON,
    data_pagamento DATETIME DEFAULT CURRENT_TIMESTAMP,
    data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (pedido_id) REFERENCES pedidos(id) ON DELETE CASCADE,
    INDEX idx_pedido_id (pedido_id),
    INDEX idx_transacao_id (transacao_id),
    INDEX idx_status (status)
);
```

15. Cupons

Armazena cupons de desconto para uso nos pedidos.

```
CREATE TABLE cupons (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo VARCHAR(50) NOT NULL UNIQUE,
    tipo ENUM('percentual', 'valor_fixo', 'frete_gratis') NOT NULL,
    valor DECIMAL(10, 2),
    data_inicio DATETIME NOT NULL,
    data_fim DATETIME NOT NULL,
    uso_maximo INT,
    uso_atual INT DEFAULT 0,
    valor_minimo DECIMAL(10, 2),
    primeira_compra BOOLEAN DEFAULT FALSE,
    status ENUM('ativo', 'inativo') DEFAULT 'ativo',
    data_cadastro DATETIME DEFAULT CURRENT_TIMESTAMP,
    data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    INDEX idx_codigo (codigo),
    INDEX idx_status (status),
    INDEX idx_data_fim (data_fim)
);
```

16. Cupom_Categorias

Relaciona cupons com categorias específicas (restrição de uso).

```
CREATE TABLE cupom_categorias (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cupom_id INT NOT NULL,
    categoria_id INT NOT NULL,
    FOREIGN KEY (cupom_id) REFERENCES cupons(id) ON DELETE CASCADE,
    FOREIGN KEY (categoria_id) REFERENCES categorias(id) ON DELETE CASCADE,
    UNIQUE KEY unique_cupom_categoria (cupom_id, categoria_id)
);
```

17. Cupom_Produtos

Relaciona cupons com produtos específicos (restrição de uso).

```
CREATE TABLE cupom_produtos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cupom_id INT NOT NULL,
  produto_id INT NOT NULL,
  FOREIGN KEY (cupom_id) REFERENCES cupons(id) ON DELETE CASCADE,
  FOREIGN KEY (produto_id) REFERENCES produtos(id) ON DELETE CASCADE,
  UNIQUE KEY unique_cupom_produto (cupom_id, produto_id)
);
```

18. Avaliações

Armazena avaliações e comentários dos usuários sobre produtos.

```
CREATE TABLE avaliacoes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  produto_id INT NOT NULL,
  usuario_id INT,
  nome VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL,
  titulo VARCHAR(100),
  comentario TEXT,
  nota TINYINT NOT NULL CHECK (nota BETWEEN 1 AND 5),
  status ENUM('pendente', 'aprovado', 'rejeitado') DEFAULT 'pendente',
  compra_verificada BOOLEAN DEFAULT FALSE,
  data_avaliacao DATETIME DEFAULT CURRENT_TIMESTAMP,
  data_aprovacao DATETIME,
  resposta_admin TEXT,
  data_resposta DATETIME,
  FOREIGN KEY (produto_id) REFERENCES produtos(id) ON DELETE CASCADE,
  FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE SET NULL,
  INDEX idx_produto_id (produto_id),
  INDEX idx_usuario_id (usuario_id),
  INDEX idx_status (status)
);
```

19. Wishlist

Armazena produtos favoritos/desejados pelos usuários.

```
CREATE TABLE wishlist (
  id INT AUTO_INCREMENT PRIMARY KEY,
  usuario_id INT NOT NULL,
```

```
    produto_id INT NOT NULL,
    data_adicao DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE,
    FOREIGN KEY (produto_id) REFERENCES produtos(id) ON DELETE CASCADE,
    UNIQUE KEY unique_usuario_produto (usuario_id, produto_id),
    INDEX idx_usuario_id (usuario_id)
);
```

20. Carrinho

Armazena itens no carrinho de compras dos usuários.

```
 CREATE TABLE carrinho (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT,
    sessao_id VARCHAR(100),
    produto_id INT NOT NULL,
    variante_id INT,
    quantidade INT NOT NULL DEFAULT 1,
    data_adicao DATETIME DEFAULT CURRENT_TIMESTAMP,
    data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE,
    FOREIGN KEY (produto_id) REFERENCES produtos(id) ON DELETE CASCADE,
    FOREIGN KEY (variante_id) REFERENCES variantes_produto(id) ON DELETE
CASCADE,
    INDEX idx_usuario_id (usuario_id),
    INDEX idx_sessao_id (sessao_id)
);
```

21. Configurações

Armazena configurações gerais da loja.

```
 CREATE TABLE configuracoes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    chave VARCHAR(100) NOT NULL UNIQUE,
    valor TEXT,
    tipo ENUM('texto', 'numero', 'boolean', 'json') DEFAULT 'texto',
    grupo VARCHAR(50) DEFAULT 'geral',
    descricao VARCHAR(255),
    data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    INDEX idx_chave (chave),
    INDEX idx_grupo (grupo)
);
```

22. Logs

Armazena logs de atividades importantes no sistema.

```
CREATE TABLE logs (
  id INT AUTO_INCREMENT PRIMARY KEY,
  usuario_id INT,
  tipo VARCHAR(50) NOT NULL,
  acao VARCHAR(50) NOT NULL,
  descricao TEXT,
  dados JSON,
  ip VARCHAR(45),
  user_agent TEXT,
  data_log DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE SET NULL,
  INDEX idx_usuario_id (usuario_id),
  INDEX idx_tipo (tipo),
  INDEX idx_data_log (data_log)
);
```

Índices e Otimizações

Para garantir a performance do banco de dados, foram definidos índices estratégicos nas tabelas:

1. **Índices de Chave Primária**: Todas as tabelas possuem uma chave primária auto-incrementada.
2. **Índices de Chave Estrangeira**: Todas as relações entre tabelas são indexadas.
3. **Índices Únicos**: Para campos que exigem unicidade, como email, SKU, slug, etc.
4. **Índices Compostos**: Para consultas frequentes que envolvem múltiplos campos.
5. **Índices de Texto**: Para campos frequentemente usados em buscas.

Estratégias de Otimização

1. **Normalização**: O esquema segue os princípios de normalização até a 3ª forma normal para minimizar redundâncias.
2. **Desnormalização Estratégica**: Em alguns casos, como em `itens_pedido`, mantemos dados redundantes (nome_produto, preco_unitario) para preservar o histórico mesmo se o produto for alterado ou excluído.
3. **Tipos de Dados Otimizados**: Escolha cuidadosa dos tipos de dados para equilibrar espaço de armazenamento e performance.
4. **JSON para Dados Flexíveis**: Uso do tipo JSON para armazenar dados com estrutura variável, como atributos de produtos e dados de pagamento.

5. **Particionamento**: Para tabelas que tendem a crescer muito, como pedidos e logs, pode-se implementar particionamento por data.

Considerações de Segurança

1. **Senhas**: Armazenadas com hash e salt (implementado na aplicação).
2. **Dados Sensíveis**: Informações de pagamento são armazenadas em formato JSON com campos sensíveis criptografados.
3. **Auditoria**: Tabela de logs para rastrear atividades importantes no sistema.
4. **Soft Delete**: Para dados críticos, considerar implementar exclusão lógica em vez de física.

Estratégia de Backup e Recuperação

1. **Backups Diários**: Implementar rotina de backup completo diário.
2. **Backups Incrementais**: A cada hora para minimizar perda de dados.
3. **Replicação**: Considerar replicação master-slave para alta disponibilidade.
4. **Testes de Recuperação**: Realizar testes periódicos de recuperação de backup.

Evolução do Esquema

O esquema foi projetado para ser flexível e permitir expansões futuras:

1. **Internacionalização**: Adicionar suporte para múltiplos idiomas.
2. **Marketplace**: Expandir para suportar múltiplos vendedores.
3. **Fidelidade**: Implementar sistema de pontos e recompensas.
4. **Assinaturas**: Adicionar suporte para produtos recorrentes.

Conclusão

A estrutura de banco de dados proposta foi projetada para atender a todas as necessidades de um e-commerce completo, seguindo as melhores práticas de design de banco de dados relacional. O esquema é robusto o suficiente para suportar as funcionalidades atuais e flexível para acomodar expansões futuras.

A implementação cuidadosa de índices, relacionamentos e tipos de dados otimizados garantirá que o sistema mantenha boa performance mesmo com o crescimento do volume de dados. As considerações de segurança e estratégias de backup asseguram a proteção e disponibilidade dos dados críticos do negócio.