Estrutura do Banco de Dados - Sistema de Compras e Cotações

Autor: Manus AI

Data: 13 de Agosto de 2025

Versão: 1.0

1. Visão Geral da Arquitetura de Dados

A estrutura do banco de dados foi projetada para suportar todas as funcionalidades do sistema de compras e cotações, garantindo integridade referencial, performance otimizada e escalabilidade. O design utiliza PostgreSQL 15+ como SGBD principal, aproveitando recursos avançados como tipos de dados JSON, full-text search, particionamento de tabelas e índices especializados.

A arquitetura de dados segue princípios de normalização até a terceira forma normal (3NF) para evitar redundâncias, com desnormalizações estratégicas em tabelas de analytics para otimizar consultas de relatórios. O schema principal será complementado por schemas específicos para auditoria, analytics e configurações do sistema.

1.1 Estratégia de Particionamento

Tabelas com grande volume de dados serão particionadas para otimizar performance e facilitar manutenção. A tabela de logs de auditoria será particionada por data (mensal), permitindo arquivamento eficiente de dados históricos. Tabelas de cotações e transações serão particionadas por ano para balancear performance de consultas recentes com retenção de histórico.

O particionamento utilizará range partitioning para dados temporais e hash partitioning para distribuição uniforme quando apropriado. Partições serão criadas automaticamente através de procedures armazenadas executadas por jobs agendados, garantindo que novas partições estejam sempre disponíveis.

1.2 Indexação Estratégica

Estratégia de indexação focará em otimizar consultas mais frequentes identificadas durante análise de requisitos. Índices compostos serão criados para consultas que filtram por múltiplas colunas simultaneamente. Índices parciais serão utilizados para otimizar consultas em subconjuntos específicos de dados, como registros ativos ou não deletados.

Índices GIN (Generalized Inverted Index) serão implementados para campos JSON e arrays, permitindo consultas eficientes em dados semi-estruturados. Índices de texto completo

(full-text search) facilitarão busca em descrições de produtos, observações de cotações e outros campos textuais.

1.3 Políticas de Retenção e Arquivamento

Dados operacionais serão mantidos online por 2 anos para acesso rápido. Dados históricos entre 2-7 anos serão movidos para partições de arquivo com acesso menos frequente. Dados com mais de 7 anos serão exportados para storage de longo prazo e removidos do banco principal, exceto quando requeridos por regulamentações específicas.

Procedures automatizadas executarão arquivamento mensal, movendo dados elegíveis para partições apropriadas. Logs de auditoria terão retenção estendida de 10 anos para compliance, com compressão automática após 1 ano para otimizar storage.

2. Schema Principal - Gestão de Usuários e Segurança

2.1 Tabela users

A tabela users armazena informações básicas de todos os usuários do sistema, servindo como entidade central para autenticação e autorização. A estrutura inclui campos obrigatórios para identificação única, credenciais seguras e metadados de controle.

SQL

```sql
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    username VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    department_id UUID REFERENCES departments(id),
    is_active BOOLEAN DEFAULT true,
    is_verified BOOLEAN DEFAULT false,
    last_login_at TIMESTAMP WITH TIME ZONE,
    password_changed_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    failed_login_attempts INTEGER DEFAULT 0,
    locked_until TIMESTAMP WITH TIME ZONE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

O campo password_hash armazenará senhas criptografadas usando bcrypt ou Argon2 com salt único por usuário. O sistema de bloqueio automático utilizará failed_login_attempts e locked_until para implementar políticas de segurança configuráveis. Timestamps de auditoria permitirão rastreamento completo de alterações.

Índices especializados incluirão índice único composto em (email, is_active) para otimizar autenticação, índice em department_id para consultas por departamento, e índice parcial em locked_until para identificação rápida de contas bloqueadas.

2.2 Tabela roles

A tabela roles define papéis disponíveis no sistema, implementando hierarquia flexível para herança de permissões. Cada role pode ter um role pai, permitindo estruturas organizacionais complexas.

SQL

```sql
CREATE TABLE roles (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT,
    parent_role_id UUID REFERENCES roles(id),
    is_system_role BOOLEAN DEFAULT false,
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

Roles do sistema (is_system_role = true) não poderão ser deletados ou modificados por usuários, garantindo integridade de permissões críticas. A hierarquia parent_role_id permitirá herança automática de permissões, simplificando gestão de acessos em organizações complexas.

2.3 Tabela permissions

A tabela permissions define permissões granulares disponíveis no sistema, organizadas por módulo e operação. Cada permissão representa uma ação específica que pode ser autorizada ou negada.

SQL

```sql
CREATE TABLE permissions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT,
    module VARCHAR(50) NOT NULL,
    operation VARCHAR(50) NOT NULL,
    resource VARCHAR(50),
    is_system_permission BOOLEAN DEFAULT false,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id)
);
```

Permissões seguirão convenção de nomenclatura module:operation:resource (ex: suppliers:read:all, quotations:approve:own). O campo resource permitirá controle granular sobre escopo de permissões, diferenciando entre acesso a próprios dados versus dados de toda organização.

2.4 Tabelas de Associação

Tabelas de associação implementarão relacionamentos many-to-many entre usuários, roles e permissões, permitindo flexibilidade máxima na configuração de acessos.

SQL

```
CREATE TABLE user_roles (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    role_id UUID NOT NULL REFERENCES roles(id) ON DELETE CASCADE,
    granted_by UUID REFERENCES users(id),
    granted_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP WITH TIME ZONE,
    is_active BOOLEAN DEFAULT true,
    UNIQUE(user_id, role_id)
);

CREATE TABLE role_permissions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    role_id UUID NOT NULL REFERENCES roles(id) ON DELETE CASCADE,
    permission_id UUID NOT NULL REFERENCES permissions(id) ON DELETE CASCADE,
    granted_by UUID REFERENCES users(id),
    granted_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    is_active BOOLEAN DEFAULT true,
    UNIQUE(role_id, permission_id)
);

CREATE TABLE user_permissions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    permission_id UUID NOT NULL REFERENCES permissions(id) ON DELETE CASCADE,
```

```
    granted_by UUID REFERENCES users(id),
    granted_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP WITH TIME ZONE,
    is_active BOOLEAN DEFAULT true,
    UNIQUE(user_id, permission_id)
);
```

A tabela user_permissions permite concessão de permissões específicas diretamente a usuários, sobrepondo permissões de roles quando necessário. Campos expires_at implementam permissões temporárias para situações especiais como delegação de autoridade.

3. Schema de Gestão Organizacional

3.1 Tabela departments

A tabela departments organiza estrutura departamental da empresa, suportando hierarquias complexas e múltiplos níveis organizacionais.

SQL

```sql
CREATE TABLE departments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    description TEXT,
    code VARCHAR(20) UNIQUE,
    parent_department_id UUID REFERENCES departments(id),
    manager_id UUID REFERENCES users(id),
    cost_center VARCHAR(50),
    budget_limit DECIMAL(15,2),
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

O campo budget_limit permitirá controle automático de gastos por departamento, com validações durante processo de aprovação de compras. A hierarquia parent_department_id suportará estruturas organizacionais complexas com múltiplos níveis de gestão.

3.2 Tabela cost_centers

A tabela cost_centers implementa centros de custo para controle financeiro detalhado, permitindo rastreamento preciso de gastos por projeto ou atividade.

SQL

```sql
CREATE TABLE cost_centers (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    code VARCHAR(50) UNIQUE NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    department_id UUID REFERENCES departments(id),
    budget_annual DECIMAL(15,2),
    budget_monthly DECIMAL(15,2),
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

Orçamentos anuais e mensais permitirão controle granular de gastos com alertas automáticos quando limites forem atingidos. Integração com departamentos facilitará consolidação de relatórios financeiros por unidade organizacional.

4. Schema de Gestão de Fornecedores

4.1 Tabela suppliers

A tabela suppliers centraliza informações completas de fornecedores, incluindo dados cadastrais, financeiros e de performance.

SQL

```sql
CREATE TABLE suppliers (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    company_name VARCHAR(255) NOT NULL,
    trade_name VARCHAR(255),
    tax_id VARCHAR(50) UNIQUE NOT NULL, -- CNPJ/CPF
    state_registration VARCHAR(50),
    municipal_registration VARCHAR(50),
    company_type VARCHAR(50), -- MEI, LTDA, SA, etc.
    business_sector VARCHAR(100),
    website VARCHAR(255),
    email VARCHAR(255),
    phone VARCHAR(20),
    mobile VARCHAR(20),
    address JSONB, -- Endereço completo estruturado
    bank_details JSONB, -- Dados bancários estruturados
```

```
    payment_terms TEXT,
    credit_limit DECIMAL(15,2),
    performance_score DECIMAL(5,2) DEFAULT 0.00,
    risk_rating VARCHAR(20) DEFAULT 'MEDIUM', -- LOW, MEDIUM, HIGH, CRITICAL
    is_active BOOLEAN DEFAULT true,
    is_approved BOOLEAN DEFAULT false,
    approved_by UUID REFERENCES users(id),
    approved_at TIMESTAMP WITH TIME ZONE,
    last_evaluation_at TIMESTAMP WITH TIME ZONE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

O campo address utilizará estrutura JSONB para flexibilidade em diferentes formatos de endereço, incluindo múltiplos endereços quando necessário. Performance_score será calculado automaticamente baseado em métricas de entrega, qualidade e relacionamento comercial.

4.2 Tabela supplier_contacts

A tabela supplier_contacts gerencia múltiplos contatos por fornecedor, permitindo comunicação direcionada por departamento ou função.

SQL

```
CREATE TABLE supplier_contacts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    supplier_id UUID NOT NULL REFERENCES suppliers(id) ON DELETE CASCADE,
    name VARCHAR(100) NOT NULL,
    position VARCHAR(100),
    department VARCHAR(100),
    email VARCHAR(255),
    phone VARCHAR(20),
    mobile VARCHAR(20),
    is_primary BOOLEAN DEFAULT false,
    is_active BOOLEAN DEFAULT true,
    notes TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

Constraint garantirá que apenas um contato por fornecedor seja marcado como primário. Sistema de comunicação utilizará contatos específicos baseado em tipo de interação (comercial, técnico, financeiro).

4.3 Tabela supplier_evaluations

A tabela supplier_evaluations registra avaliações periódicas de fornecedores, alimentando sistema de scoring automático.

SQL

```
CREATE TABLE supplier_evaluations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    supplier_id UUID NOT NULL REFERENCES suppliers(id) ON DELETE CASCADE,
    evaluator_id UUID NOT NULL REFERENCES users(id),
    evaluation_period_start DATE NOT NULL,
    evaluation_period_end DATE NOT NULL,
    quality_score DECIMAL(3,2) CHECK (quality_score >= 0 AND quality_score
<= 10),
    delivery_score DECIMAL(3,2) CHECK (delivery_score >= 0 AND
delivery_score <= 10),
    service_score DECIMAL(3,2) CHECK (service_score >= 0 AND service_score
<= 10),
    price_competitiveness DECIMAL(3,2) CHECK (price_competitiveness >= 0 AND
price_competitiveness <= 10),
    overall_score DECIMAL(3,2) GENERATED ALWAYS AS (
        (quality_score + delivery_score + service_score +
price_competitiveness) / 4
    ) STORED,
    comments TEXT,
    recommendations TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id)
);
```

Overall_score será calculado automaticamente como média ponderada dos critérios individuais. Histórico de avaliações permitirá análise de tendências e identificação de fornecedores em melhoria ou declínio.

4.4 Tabela supplier_documents

A tabela supplier_documents gerencia documentos e certificações de fornecedores, com controle de validade e renovação automática.

SQL

```sql
CREATE TABLE supplier_documents (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    supplier_id UUID NOT NULL REFERENCES suppliers(id) ON DELETE CASCADE,
    document_type VARCHAR(100) NOT NULL, -- Contrato Social, Certidões, etc.
    document_number VARCHAR(100),
    file_path VARCHAR(500),
    file_name VARCHAR(255),
    file_size INTEGER,
    mime_type VARCHAR(100),
    issued_date DATE,
    expiry_date DATE,
    is_valid BOOLEAN DEFAULT true,
    verification_status VARCHAR(50) DEFAULT 'PENDING', -- PENDING, VERIFIED,
REJECTED
    verified_by UUID REFERENCES users(id),
    verified_at TIMESTAMP WITH TIME ZONE,
    notes TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

Sistema de alertas monitorará expiry_date para notificar renovações necessárias com antecedência configurável. Verificação manual de documentos críticos garantirá compliance com políticas internas.

5. Schema de Gestão de Produtos e Serviços

5.1 Tabela product_categories

A tabela product_categories organiza produtos em hierarquia de categorias, facilitando busca e relatórios agrupados.

SQL

```sql
CREATE TABLE product_categories (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    description TEXT,
    parent_category_id UUID REFERENCES product_categories(id),
    category_path TEXT, -- Path completo para busca eficiente
    level INTEGER DEFAULT 0,
    sort_order INTEGER DEFAULT 0,
    is_active BOOLEAN DEFAULT true,
```

```
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

O campo category_path armazenará caminho completo da categoria (ex: "Materiais > Elétricos > Cabos") para otimizar consultas hierárquicas. Triggers automáticos manterão consistência do path quando hierarquia for alterada.

5.2 Tabela products

A tabela products centraliza catálogo completo de produtos e serviços, com especificações técnicas flexíveis.

SQL

```sql
CREATE TABLE products (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    code VARCHAR(100) UNIQUE,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    category_id UUID REFERENCES product_categories(id),
    unit_of_measure VARCHAR(20) NOT NULL, -- UN, KG, M, L, etc.
    specifications JSONB, -- Especificações técnicas flexíveis
    brand VARCHAR(100),
    model VARCHAR(100),
    manufacturer VARCHAR(100),
    minimum_order_quantity DECIMAL(10,3) DEFAULT 1,
    maximum_order_quantity DECIMAL(10,3),
    lead_time_days INTEGER,
    shelf_life_days INTEGER,
    storage_requirements TEXT,
    safety_notes TEXT,
    is_active BOOLEAN DEFAULT true,
    is_service BOOLEAN DEFAULT false,
    requires_approval BOOLEAN DEFAULT false,
    search_vector tsvector, -- Para full-text search
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

O campo specifications utilizará JSONB para armazenar especificações técnicas de forma flexível, permitindo diferentes estruturas para diferentes tipos de produtos. Search_vector

será mantido automaticamente via triggers para busca full-text eficiente.

5.3 Tabela product_suppliers

A tabela product_suppliers vincula produtos a fornecedores com informações específicas de fornecimento.

SQL

```sql
CREATE TABLE product_suppliers (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    product_id UUID NOT NULL REFERENCES products(id) ON DELETE CASCADE,
    supplier_id UUID NOT NULL REFERENCES suppliers(id) ON DELETE CASCADE,
    supplier_product_code VARCHAR(100),
    supplier_product_name VARCHAR(255),
    current_price DECIMAL(15,4),
    currency VARCHAR(3) DEFAULT 'BRL',
    minimum_order_quantity DECIMAL(10,3),
    lead_time_days INTEGER,
    is_preferred BOOLEAN DEFAULT false,
    is_active BOOLEAN DEFAULT true,
    last_price_update TIMESTAMP WITH TIME ZONE,
    price_valid_until DATE,
    notes TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id),
    UNIQUE(product_id, supplier_id)
);
```

Histórico de preços será mantido em tabela separada para análise de tendências. Campo is_preferred permitirá marcação de fornecedores preferenciais por produto, influenciando sugestões automáticas do sistema.

5.4 Tabela product_price_history

A tabela product_price_history mantém histórico completo de preços para análise de tendências e flutuações.

SQL

```sql
CREATE TABLE product_price_history (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    product_supplier_id UUID NOT NULL REFERENCES product_suppliers(id) ON
DELETE CASCADE,
    price DECIMAL(15,4) NOT NULL,
```

```
    currency VARCHAR(3) DEFAULT 'BRL',
    effective_date DATE NOT NULL,
    end_date DATE,
    price_source VARCHAR(50), -- QUOTATION, MANUAL, IMPORT, API
    quotation_id UUID REFERENCES quotations(id),
    notes TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id)
) PARTITION BY RANGE (effective_date);
```

Tabela será particionada por ano para otimizar consultas de histórico recente. Triggers automáticos atualizarão end_date quando novos preços forem inseridos, mantendo integridade temporal.

6. Schema de Processo de Compras

6.1 Tabela requisitions

A tabela requisitions gerencia solicitações de compra com workflow completo de aprovações.

SQL

```sql
CREATE TABLE requisitions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    number VARCHAR(50) UNIQUE NOT NULL, -- Numeração sequencial
    title VARCHAR(255) NOT NULL,
    description TEXT,
    requester_id UUID NOT NULL REFERENCES users(id),
    department_id UUID NOT NULL REFERENCES departments(id),
    cost_center_id UUID REFERENCES cost_centers(id),
    priority VARCHAR(20) DEFAULT 'MEDIUM', -- LOW, MEDIUM, HIGH, URGENT
    status VARCHAR(50) DEFAULT 'DRAFT', -- DRAFT, SUBMITTED, APPROVED,
REJECTED, CANCELLED
    total_estimated_value DECIMAL(15,2),
    currency VARCHAR(3) DEFAULT 'BRL',
    justification TEXT NOT NULL,
    delivery_location TEXT,
    required_delivery_date DATE,
    budget_approval_required BOOLEAN DEFAULT false,
    technical_approval_required BOOLEAN DEFAULT false,
    workflow_id UUID REFERENCES workflows(id),
    current_step INTEGER DEFAULT 1,
    submitted_at TIMESTAMP WITH TIME ZONE,
    approved_at TIMESTAMP WITH TIME ZONE,
    rejected_at TIMESTAMP WITH TIME ZONE,
```

```
    rejection_reason TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

Numeração sequencial será gerada automaticamente por ano (ex: REQ-2025-0001). Workflow_id permitirá diferentes fluxos de aprovação baseados em valor, departamento ou tipo de compra.

6.2 Tabela requisition_items

A tabela requisition_items detalha itens solicitados em cada requisição.

SQL

```
CREATE TABLE requisition_items (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    requisition_id UUID NOT NULL REFERENCES requisitions(id) ON DELETE
CASCADE,
    product_id UUID REFERENCES products(id),
    custom_description TEXT, -- Para produtos não cadastrados
    quantity DECIMAL(10,3) NOT NULL,
    unit_of_measure VARCHAR(20) NOT NULL,
    estimated_unit_price DECIMAL(15,4),
    estimated_total_price DECIMAL(15,2),
    currency VARCHAR(3) DEFAULT 'BRL',
    specifications JSONB, -- Especificações específicas do item
    technical_requirements TEXT,
    delivery_date DATE,
    notes TEXT,
    line_number INTEGER NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id),
    UNIQUE(requisition_id, line_number)
);
```

Campo custom_description permitirá solicitação de produtos não cadastrados, que serão posteriormente adicionados ao catálogo. Specifications em JSONB oferecerá flexibilidade para requisitos específicos por item.

6.3 Tabela quotations

A tabela quotations gerencia processo completo de cotações com múltiplos fornecedores.

SQL

```
CREATE TABLE quotations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    number VARCHAR(50) UNIQUE NOT NULL, -- Numeração sequencial
    title VARCHAR(255) NOT NULL,
    description TEXT,
    requisition_id UUID REFERENCES requisitions(id),
    quotation_manager_id UUID NOT NULL REFERENCES users(id),
    status VARCHAR(50) DEFAULT 'DRAFT', -- DRAFT, SENT, RECEIVING,
ANALYZING, DECIDED, CANCELLED
    quotation_type VARCHAR(50) DEFAULT 'STANDARD', -- STANDARD, EMERGENCY,
FRAMEWORK
    total_estimated_value DECIMAL(15,2),
    currency VARCHAR(3) DEFAULT 'BRL',
    response_deadline DATE,
    evaluation_criteria JSONB, -- Critérios e pesos para avaliação
    terms_and_conditions TEXT,
    delivery_location TEXT,
    delivery_deadline DATE,
    payment_terms TEXT,
    sent_at TIMESTAMP WITH TIME ZONE,
    decision_made_at TIMESTAMP WITH TIME ZONE,
    winning_supplier_id UUID REFERENCES suppliers(id),
    decision_justification TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

Evaluation_criteria em JSONB permitirá configuração flexível de critérios de avaliação com pesos específicos (preço 40%, prazo 30%, qualidade 30%, etc.). Sistema calculará automaticamente scores baseados nesses critérios.

6.4 Tabela quotation_items

A tabela quotation_items detalha itens incluídos em cada cotação.

SQL

```
CREATE TABLE quotation_items (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    quotation_id UUID NOT NULL REFERENCES quotations(id) ON DELETE CASCADE,
    requisition_item_id UUID REFERENCES requisition_items(id),
```

```
    product_id UUID REFERENCES products(id),
    description TEXT NOT NULL,
    quantity DECIMAL(10,3) NOT NULL,
    unit_of_measure VARCHAR(20) NOT NULL,
    specifications JSONB,
    technical_requirements TEXT,
    delivery_date DATE,
    line_number INTEGER NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    UNIQUE(quotation_id, line_number)
);
```

Vinculação com requisition_item_id manterá rastreabilidade entre solicitação original e cotação. Especificações poderão ser refinadas durante processo de cotação.

6.5 Tabela quotation_suppliers

A tabela quotation_suppliers gerencia fornecedores convidados para cada cotação.

SQL

```
CREATE TABLE quotation_suppliers (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    quotation_id UUID NOT NULL REFERENCES quotations(id) ON DELETE CASCADE,
    supplier_id UUID NOT NULL REFERENCES suppliers(id),
    invitation_sent_at TIMESTAMP WITH TIME ZONE,
    invitation_method VARCHAR(50), -- EMAIL, WHATSAPP, PHONE, PORTAL
    response_received_at TIMESTAMP WITH TIME ZONE,
    response_status VARCHAR(50) DEFAULT 'PENDING', -- PENDING, RESPONDED,
DECLINED, NO_RESPONSE
    total_quoted_value DECIMAL(15,2),
    currency VARCHAR(3) DEFAULT 'BRL',
    delivery_days INTEGER,
    payment_terms TEXT,
    validity_days INTEGER,
    observations TEXT,
    is_winner BOOLEAN DEFAULT false,
    evaluation_score DECIMAL(5,2),
    evaluation_notes TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id),
    UNIQUE(quotation_id, supplier_id)
);
```

Sistema de scoring automático calculará evaluation_score baseado em critérios configurados na cotação. Histórico de respostas influenciará futuras seleções de fornecedores.

6.6 Tabela quotation_responses

A tabela quotation_responses armazena propostas detalhadas de cada fornecedor.

SQL

```
CREATE TABLE quotation_responses (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    quotation_supplier_id UUID NOT NULL REFERENCES quotation_suppliers(id)
ON DELETE CASCADE,
    quotation_item_id UUID NOT NULL REFERENCES quotation_items(id),
    unit_price DECIMAL(15,4) NOT NULL,
    total_price DECIMAL(15,2) NOT NULL,
    currency VARCHAR(3) DEFAULT 'BRL',
    delivery_days INTEGER,
    minimum_quantity DECIMAL(10,3),
    brand VARCHAR(100),
    model VARCHAR(100),
    specifications_compliance BOOLEAN DEFAULT true,
    alternative_specifications TEXT,
    observations TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id),
    UNIQUE(quotation_supplier_id, quotation_item_id)
);
```

Campo specifications_compliance indicará se proposta atende completamente às especificações solicitadas. Alternative_specifications permitirá fornecedores sugerirem alternativas quando especificações exatas não estiverem disponíveis.

7. Schema de Comunicação e Notificações

7.1 Tabela communications

A tabela communications centraliza todo histórico de comunicações do sistema, incluindo emails, WhatsApp e notificações internas.

SQL

```sql
CREATE TABLE communications (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    type VARCHAR(50) NOT NULL, -- EMAIL, WHATSAPP, SMS, NOTIFICATION, CHAT
    direction VARCHAR(20) NOT NULL, -- OUTBOUND, INBOUND
    sender_id UUID REFERENCES users(id),
    recipient_type VARCHAR(50) NOT NULL, -- USER, SUPPLIER, EXTERNAL
    recipient_id UUID, -- user_id ou supplier_id dependendo do tipo
    recipient_address VARCHAR(255), -- email, phone, etc.
    subject VARCHAR(500),
    content TEXT,
    template_id UUID REFERENCES communication_templates(id),
    status VARCHAR(50) DEFAULT 'PENDING', -- PENDING, SENT, DELIVERED, READ,
FAILED
    sent_at TIMESTAMP WITH TIME ZONE,
    delivered_at TIMESTAMP WITH TIME ZONE,
    read_at TIMESTAMP WITH TIME ZONE,
    error_message TEXT,
    metadata JSONB, -- Dados específicos do canal (message_id, tracking,
etc.)
    related_entity_type VARCHAR(50), -- QUOTATION, REQUISITION, SUPPLIER,
etc.
    related_entity_id UUID,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id)
) PARTITION BY RANGE (created_at);
```

Particionamento mensal otimizará consultas de histórico recente. Metadata em JSONB armazenará informações específicas de cada canal como IDs de mensagem, status de entrega e dados de tracking.

7.2 Tabela communication_templates

A tabela communication_templates gerencia templates reutilizáveis para diferentes tipos de comunicação.

SQL

```sql
CREATE TABLE communication_templates (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    description TEXT,
    type VARCHAR(50) NOT NULL, -- EMAIL, WHATSAPP, SMS, NOTIFICATION
    category VARCHAR(100), -- QUOTATION_REQUEST, APPROVAL_NEEDED, etc.
    subject_template TEXT,
    content_template TEXT NOT NULL,
    variables JSONB, -- Variáveis disponíveis no template
```

```
    is_system_template BOOLEAN DEFAULT false,
    is_active BOOLEAN DEFAULT true,
    language VARCHAR(5) DEFAULT 'pt-BR',
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

Templates utilizarão sintaxe Jinja2 para substituição de variáveis. Sistema de versionamento permitirá testes A/B e rollback de templates quando necessário.

7.3 Tabela notifications

A tabela notifications gerencia notificações internas do sistema para usuários.

SQL

```sql
CREATE TABLE notifications (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    type VARCHAR(50) NOT NULL, -- INFO, WARNING, ERROR, SUCCESS
    category VARCHAR(100), -- APPROVAL_NEEDED, QUOTATION_RECEIVED, etc.
    title VARCHAR(255) NOT NULL,
    message TEXT NOT NULL,
    action_url VARCHAR(500), -- URL para ação relacionada
    action_label VARCHAR(100), -- Label do botão de ação
    is_read BOOLEAN DEFAULT false,
    read_at TIMESTAMP WITH TIME ZONE,
    priority VARCHAR(20) DEFAULT 'NORMAL', -- LOW, NORMAL, HIGH, URGENT
    expires_at TIMESTAMP WITH TIME ZONE,
    related_entity_type VARCHAR(50),
    related_entity_id UUID,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
) PARTITION BY RANGE (created_at);
```

Particionamento mensal com retenção automática de 6 meses para notificações lidas. Sistema de expiração automática removerá notificações antigas irrelevantes.

7.4 Tabela chat_conversations

A tabela chat_conversations gerencia conversas do chat interno entre usuários.

SQL

```sql
CREATE TABLE chat_conversations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
    title VARCHAR(255),
    type VARCHAR(50) DEFAULT 'DIRECT', -- DIRECT, GROUP, CHANNEL
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id)
);

CREATE TABLE chat_participants (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    conversation_id UUID NOT NULL REFERENCES chat_conversations(id) ON
DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    role VARCHAR(50) DEFAULT 'MEMBER', -- ADMIN, MODERATOR, MEMBER
    joined_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    left_at TIMESTAMP WITH TIME ZONE,
    is_active BOOLEAN DEFAULT true,
    UNIQUE(conversation_id, user_id)
);

CREATE TABLE chat_messages (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    conversation_id UUID NOT NULL REFERENCES chat_conversations(id) ON
DELETE CASCADE,
    sender_id UUID NOT NULL REFERENCES users(id),
    message_type VARCHAR(50) DEFAULT 'TEXT', -- TEXT, FILE, IMAGE, SYSTEM
    content TEXT,
    file_path VARCHAR(500),
    file_name VARCHAR(255),
    file_size INTEGER,
    mime_type VARCHAR(100),
    reply_to_id UUID REFERENCES chat_messages(id),
    is_edited BOOLEAN DEFAULT false,
    edited_at TIMESTAMP WITH TIME ZONE,
    is_deleted BOOLEAN DEFAULT false,
    deleted_at TIMESTAMP WITH TIME ZONE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
) PARTITION BY RANGE (created_at);
```

Sistema de busca full-text permitirá localização rápida de mensagens históricas. Suporte a threads via reply_to_id facilitará organização de conversas complexas.

8. Schema de Analytics e Relatórios

8.1 Tabela analytics_dashboards

A tabela analytics_dashboards gerencia dashboards personalizáveis por usuário.

SQL

```sql
CREATE TABLE analytics_dashboards (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    description TEXT,
    owner_id UUID NOT NULL REFERENCES users(id),
    is_public BOOLEAN DEFAULT false,
    is_default BOOLEAN DEFAULT false,
    layout JSONB NOT NULL, -- Configuração de widgets e layout
    filters JSONB, -- Filtros padrão do dashboard
    refresh_interval INTEGER DEFAULT 300, -- Segundos
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

Layout em JSONB armazenará configuração completa de widgets, posições e tamanhos. Sistema de compartilhamento permitirá dashboards públicos para toda organização.

8.2 Tabela analytics_widgets

A tabela analytics_widgets define widgets disponíveis para composição de dashboards.

SQL

```sql
CREATE TABLE analytics_widgets (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    description TEXT,
    type VARCHAR(50) NOT NULL, -- CHART, TABLE, KPI, GAUGE, etc.
    category VARCHAR(100), -- FINANCIAL, OPERATIONAL, PERFORMANCE, etc.
    query_template TEXT NOT NULL, -- SQL template para dados
    chart_config JSONB, -- Configuração específica do gráfico
    default_filters JSONB,
    required_permissions TEXT[], -- Permissões necessárias
    is_system_widget BOOLEAN DEFAULT false,
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

Query_template utilizará sintaxe parametrizada para substituição segura de filtros. Chart_config definirá tipo de gráfico, cores, eixos e outras configurações visuais.

8.3 Tabela analytics_reports

A tabela analytics_reports gerencia relatórios agendados e sob demanda.

SQL

```sql
CREATE TABLE analytics_reports (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    description TEXT,
    type VARCHAR(50) NOT NULL, -- SCHEDULED, ON_DEMAND, AUTOMATED
    category VARCHAR(100),
    template_id UUID REFERENCES report_templates(id),
    owner_id UUID NOT NULL REFERENCES users(id),
    parameters JSONB, -- Parâmetros do relatório
    schedule_cron VARCHAR(100), -- Expressão cron para agendamento
    output_format VARCHAR(20) DEFAULT 'PDF', -- PDF, EXCEL, CSV
    recipients JSONB, -- Lista de destinatários
    last_run_at TIMESTAMP WITH TIME ZONE,
    next_run_at TIMESTAMP WITH TIME ZONE,
    status VARCHAR(50) DEFAULT 'ACTIVE', -- ACTIVE, PAUSED, COMPLETED, ERROR
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);
```

Sistema de agendamento utilizará Celery para execução de relatórios em background. Histórico de execuções será mantido para troubleshooting e auditoria.

8.4 Tabela analytics_kpis

A tabela analytics_kpis define e rastreia indicadores-chave de performance.

SQL

```sql
CREATE TABLE analytics_kpis (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    description TEXT,
    category VARCHAR(100),
    calculation_method TEXT NOT NULL, -- SQL ou fórmula
    unit VARCHAR(20), -- %, BRL, DAYS, etc.
```

```
    target_value DECIMAL(15,4),
    warning_threshold DECIMAL(15,4),
    critical_threshold DECIMAL(15,4),
    trend_direction VARCHAR(20) DEFAULT 'HIGHER_BETTER', -- HIGHER_BETTER,
LOWER_BETTER
    calculation_frequency VARCHAR(50) DEFAULT 'DAILY', -- REAL_TIME, HOURLY,
DAILY, WEEKLY
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);

CREATE TABLE analytics_kpi_values (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    kpi_id UUID NOT NULL REFERENCES analytics_kpis(id) ON DELETE CASCADE,
    period_start TIMESTAMP WITH TIME ZONE NOT NULL,
    period_end TIMESTAMP WITH TIME ZONE NOT NULL,
    value DECIMAL(15,4) NOT NULL,
    target_value DECIMAL(15,4),
    variance_percentage DECIMAL(5,2),
    status VARCHAR(50), -- ON_TARGET, WARNING, CRITICAL
    calculated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(kpi_id, period_start, period_end)
) PARTITION BY RANGE (period_start);
```

Valores de KPI serão calculados automaticamente por jobs agendados. Sistema de alertas notificará quando valores estiverem fora dos thresholds definidos.

9. Schema de Auditoria e Compliance

9.1 Tabela audit_logs

A tabela audit_logs registra todas as operações do sistema para rastreabilidade completa.

SQL

```
CREATE TABLE audit_logs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id),
    session_id VARCHAR(255),
    ip_address INET,
    user_agent TEXT,
    action VARCHAR(100) NOT NULL, -- CREATE, UPDATE, DELETE, LOGIN, LOGOUT,
etc.
```

```
    entity_type VARCHAR(100), -- TABLE ou ENTITY afetada
    entity_id UUID,
    old_values JSONB, -- Valores antes da alteração
    new_values JSONB, -- Valores após a alteração
    changed_fields TEXT[], -- Campos que foram alterados
    success BOOLEAN DEFAULT true,
    error_message TEXT,
    request_id VARCHAR(100), -- Para correlação de requests
    duration_ms INTEGER, -- Duração da operação
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
) PARTITION BY RANGE (created_at);
```

Particionamento mensal com retenção de 10 anos para compliance. Índices especializados em user_id, entity_type e created_at otimizarão consultas de auditoria.

9.2 Tabela security_events

A tabela security_events registra eventos de segurança para monitoramento e análise.

SQL

```
CREATE TABLE security_events (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    event_type VARCHAR(100) NOT NULL, -- LOGIN_FAILED, PERMISSION_DENIED,
etc.
    severity VARCHAR(20) NOT NULL, -- LOW, MEDIUM, HIGH, CRITICAL
    user_id UUID REFERENCES users(id),
    ip_address INET,
    user_agent TEXT,
    description TEXT NOT NULL,
    details JSONB, -- Detalhes específicos do evento
    source_system VARCHAR(100), -- Sistema que gerou o evento
    is_resolved BOOLEAN DEFAULT false,
    resolved_by UUID REFERENCES users(id),
    resolved_at TIMESTAMP WITH TIME ZONE,
    resolution_notes TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
) PARTITION BY RANGE (created_at);
```

Sistema de correlação identificará padrões suspeitos como múltiplas tentativas de login falhadas ou acessos anômalos. Alertas automáticos serão gerados para eventos críticos.

9.3 Tabela compliance_checks

A tabela compliance_checks registra verificações automáticas de conformidade.

SQL

```sql
CREATE TABLE compliance_checks (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    check_name VARCHAR(100) NOT NULL,
    check_type VARCHAR(50) NOT NULL, -- LGPD, SECURITY, BUSINESS_RULE, etc.
    description TEXT,
    check_query TEXT, -- Query para verificação
    expected_result JSONB, -- Resultado esperado
    actual_result JSONB, -- Resultado obtido
    status VARCHAR(50) NOT NULL, -- PASSED, FAILED, WARNING, SKIPPED
    severity VARCHAR(20) DEFAULT 'MEDIUM',
    remediation_steps TEXT,
    next_check_at TIMESTAMP WITH TIME ZONE,
    check_frequency VARCHAR(50), -- DAILY, WEEKLY, MONTHLY
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
) PARTITION BY RANGE (created_at);
```

Checks automáticos verificarão conformidade com LGPD, políticas de segurança e regras de negócio. Relatórios de compliance serão gerados automaticamente baseados nos resultados.

10. Schema de Configuração e Sistema

10.1 Tabela system_settings

A tabela system_settings centraliza configurações globais do sistema.

SQL

```sql
CREATE TABLE system_settings (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    category VARCHAR(100) NOT NULL,
    key VARCHAR(100) NOT NULL,
    value TEXT,
    data_type VARCHAR(50) DEFAULT 'STRING', -- STRING, INTEGER, BOOLEAN, JSON
    description TEXT,
    is_encrypted BOOLEAN DEFAULT false,
    is_system_setting BOOLEAN DEFAULT false,
    validation_rule TEXT, -- Regex ou regra de validação
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_by UUID REFERENCES users(id),
    UNIQUE(category, key)
);
```

Configurações sensíveis serão criptografadas automaticamente. Sistema de validação garantirá que alterações não quebrem funcionalidades críticas.

10.2 Tabela workflows

A tabela workflows define fluxos de aprovação configuráveis.

SQL

```sql
CREATE TABLE workflows (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    description TEXT,
    entity_type VARCHAR(100) NOT NULL, -- REQUISITION, QUOTATION, etc.
    conditions JSONB, -- Condições para aplicação do workflow
    is_active BOOLEAN DEFAULT true,
    is_default BOOLEAN DEFAULT false,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id)
);

CREATE TABLE workflow_steps (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    workflow_id UUID NOT NULL REFERENCES workflows(id) ON DELETE CASCADE,
    step_number INTEGER NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    step_type VARCHAR(50) NOT NULL, -- APPROVAL, NOTIFICATION, AUTOMATION
    approver_type VARCHAR(50), -- USER, ROLE, DEPARTMENT, DYNAMIC
    approver_id UUID, -- user_id, role_id, department_id
    approval_criteria JSONB, -- Critérios específicos de aprovação
    timeout_hours INTEGER,
    escalation_to UUID REFERENCES users(id),
    is_parallel BOOLEAN DEFAULT false, -- Aprovação paralela
    is_optional BOOLEAN DEFAULT false,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(workflow_id, step_number)
);
```

Workflows suportarão aprovações paralelas, escalação automática e critérios dinâmicos baseados em valor, departamento ou outras variáveis.

10.3 Tabela workflow_instances

A tabela workflow_instances rastreia execução de workflows para cada entidade.

SQL

```sql
CREATE TABLE workflow_instances (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    workflow_id UUID NOT NULL REFERENCES workflows(id),
    entity_type VARCHAR(100) NOT NULL,
    entity_id UUID NOT NULL,
    current_step INTEGER DEFAULT 1,
    status VARCHAR(50) DEFAULT 'IN_PROGRESS', -- IN_PROGRESS, COMPLETED,
REJECTED, CANCELLED
    started_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    completed_at TIMESTAMP WITH TIME ZONE,
    started_by UUID REFERENCES users(id),
    UNIQUE(entity_type, entity_id)
);

CREATE TABLE workflow_step_executions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    workflow_instance_id UUID NOT NULL REFERENCES workflow_instances(id) ON
DELETE CASCADE,
    workflow_step_id UUID NOT NULL REFERENCES workflow_steps(id),
    status VARCHAR(50) DEFAULT 'PENDING', -- PENDING, APPROVED, REJECTED,
SKIPPED, TIMEOUT
    assigned_to UUID REFERENCES users(id),
    started_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    completed_at TIMESTAMP WITH TIME ZONE,
    comments TEXT,
    decision VARCHAR(50), -- APPROVE, REJECT, DELEGATE
    delegated_to UUID REFERENCES users(id),
    timeout_at TIMESTAMP WITH TIME ZONE,
    UNIQUE(workflow_instance_id, workflow_step_id)
);
```

Sistema de timeout automático escalará aprovações pendentes conforme configurado. Histórico completo de execuções permitirá análise de gargalos e otimização de processos.

11. Índices e Otimizações

11.1 Índices Principais

SQL

```sql
-- Índices de autenticação e autorização
CREATE INDEX idx_users_email_active ON users(email, is_active);
CREATE INDEX idx_users_department ON users(department_id) WHERE is_active =
```

```
true;
CREATE INDEX idx_user_roles_user ON user_roles(user_id) WHERE is_active =
true;

-- Índices de fornecedores
CREATE INDEX idx_suppliers_performance ON suppliers(performance_score DESC,
is_active);
CREATE INDEX idx_suppliers_tax_id ON suppliers(tax_id) WHERE is_active =
true;
CREATE INDEX idx_supplier_contacts_supplier ON
supplier_contacts(supplier_id) WHERE is_active = true;

-- Índices de produtos
CREATE INDEX idx_products_category ON products(category_id) WHERE is_active
= true;
CREATE INDEX idx_products_search ON products USING gin(search_vector);
CREATE INDEX idx_product_suppliers_product ON product_suppliers(product_id)
WHERE is_active = true;

-- Índices de cotações
CREATE INDEX idx_requisitions_status_date ON requisitions(status, created_at
DESC);
CREATE INDEX idx_quotations_status_date ON quotations(status, created_at
DESC);
CREATE INDEX idx_quotation_responses_quotation ON
quotation_responses(quotation_supplier_id);

-- Índices de comunicação
CREATE INDEX idx_communications_recipient ON communications(recipient_type,
recipient_id, created_at DESC);
CREATE INDEX idx_notifications_user_unread ON notifications(user_id,
is_read, created_at DESC);

-- Índices de auditoria
CREATE INDEX idx_audit_logs_user_date ON audit_logs(user_id, created_at
DESC);
CREATE INDEX idx_audit_logs_entity ON audit_logs(entity_type, entity_id,
created_at DESC);
```

11.2 Views Materializadas

SQL

```
-- View para dashboard de fornecedores
CREATE MATERIALIZED VIEW mv_supplier_performance AS
SELECT
    s.id,
```

```
    s.company_name,
    s.performance_score,
    COUNT(DISTINCT qs.quotation_id) as quotations_count,
    COUNT(DISTINCT CASE WHEN qs.is_winner THEN qs.quotation_id END) as
wins_count,
    AVG(qr.unit_price) as avg_unit_price,
    MAX(qs.updated_at) as last_quotation_date
FROM suppliers s
LEFT JOIN quotation_suppliers qs ON s.id = qs.supplier_id
LEFT JOIN quotation_responses qr ON qs.id = qr.quotation_supplier_id
WHERE s.is_active = true
GROUP BY s.id, s.company_name, s.performance_score;

CREATE UNIQUE INDEX idx_mv_supplier_performance_id ON
mv_supplier_performance(id);

-- View para analytics de gastos
CREATE MATERIALIZED VIEW mv_spending_analytics AS
SELECT
    DATE_TRUNC('month', r.created_at) as month,
    d.name as department_name,
    cc.name as cost_center_name,
    SUM(r.total_estimated_value) as total_requested,
    COUNT(*) as requisitions_count,
    AVG(r.total_estimated_value) as avg_requisition_value
FROM requisitions r
JOIN departments d ON r.department_id = d.id
LEFT JOIN cost_centers cc ON r.cost_center_id = cc.id
WHERE r.status IN ('APPROVED', 'COMPLETED')
GROUP BY DATE_TRUNC('month', r.created_at), d.name, cc.name;

CREATE INDEX idx_mv_spending_analytics_month ON mv_spending_analytics(month
DESC);
```

11.3 Triggers e Procedures

SQL

```
-- Trigger para atualização automática de search_vector em produtos
CREATE OR REPLACE FUNCTION update_product_search_vector()
RETURNS TRIGGER AS $$
BEGIN
    NEW.search_vector := to_tsvector('portuguese',
        COALESCE(NEW.name, '') || ' ' ||
        COALESCE(NEW.description, '') || ' ' ||
        COALESCE(NEW.brand, '') || ' ' ||
        COALESCE(NEW.model, '')
```

```
    );
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_update_product_search_vector
    BEFORE INSERT OR UPDATE ON products
    FOR EACH ROW EXECUTE FUNCTION update_product_search_vector();

-- Trigger para auditoria automática
CREATE OR REPLACE FUNCTION audit_trigger_function()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO audit_logs (action, entity_type, entity_id, new_values,
user_id)
        VALUES ('CREATE', TG_TABLE_NAME, NEW.id, to_jsonb(NEW),
NEW.created_by);
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO audit_logs (action, entity_type, entity_id, old_values,
new_values, user_id)
        VALUES ('UPDATE', TG_TABLE_NAME, NEW.id, to_jsonb(OLD),
to_jsonb(NEW), NEW.updated_by);
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO audit_logs (action, entity_type, entity_id, old_values,
user_id)
        VALUES ('DELETE', TG_TABLE_NAME, OLD.id, to_jsonb(OLD),
OLD.updated_by);
        RETURN OLD;
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

-- Procedure para refresh automático de views materializadas
CREATE OR REPLACE FUNCTION refresh_materialized_views()
RETURNS void AS $$
BEGIN
    REFRESH MATERIALIZED VIEW CONCURRENTLY mv_supplier_performance;
    REFRESH MATERIALIZED VIEW CONCURRENTLY mv_spending_analytics;
END;
$$ LANGUAGE plpgsql;
```

12. Estratégias de Backup e Manutenção

12.1 Políticas de Backup

O sistema implementará backup automático em múltiplas camadas para garantir recuperação completa em diferentes cenários. Backups completos serão executados semanalmente aos domingos durante janela de manutenção, com backups incrementais diários para capturar alterações desde o último backup completo.

Backups de transação log serão executados a cada 15 minutos para minimizar perda de dados em caso de falha. Todos os backups serão criptografados usando AES-256 e armazenados em múltiplas regiões geográficas para proteção contra desastres regionais.

12.2 Manutenção Automática

Jobs de manutenção automática serão agendados para execução durante horários de baixa utilização. Reindexação automática será executada mensalmente para otimizar performance de consultas. Análise de estatísticas de tabelas será executada semanalmente para manter planos de execução otimizados.

Limpeza automática de dados temporários incluirá remoção de sessões expiradas, logs antigos conforme políticas de retenção, e arquivamento de dados históricos. Monitoramento contínuo de crescimento de tabelas alertará para necessidade de particionamento adicional.

12.3 Monitoramento de Performance

Sistema de monitoramento contínuo rastreará métricas críticas de performance incluindo tempo de resposta de consultas, utilização de CPU e memória, e crescimento de dados. Alertas automáticos serão configurados para degradação de performance ou utilização excessiva de recursos.

Análise de consultas lentas identificará oportunidades de otimização através de novos índices ou reescrita de queries. Relatórios semanais de performance fornecerão insights para planejamento de capacidade e otimizações futuras.

Esta estrutura de banco de dados robusta e bem otimizada fornecerá a base sólida necessária para suportar todas as funcionalidades do sistema de compras e cotações, garantindo performance, escalabilidade e integridade dos dados em todas as operações.