StepFlow Gateway 数据库设计

概述

设计一个完整的数据库结构来支持：

1. OpenAPI 文档模板存储
2. 解析后的 API 文档存储
3. 前端展示支持
4. HTTP 执行支持
5. 工作流节点和调度任务引用

数据库表设计

1. OpenAPI 模板表 (openapi_templates)

```
CREATE TABLE openapi_templates (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL,
    description TEXT,
    version VARCHAR(50) NOT NULL,
    category VARCHAR(100), -- 分类：如 "电商", "支付", "用户管理" 等
    tags TEXT[], -- 标签数组
    template_content TEXT NOT NULL, -- 原始 OpenAPI 文档内容
    content_type VARCHAR(20) NOT NULL DEFAULT 'yaml', -- 'yaml' 或
'json'
    is_public BOOLEAN DEFAULT false, -- 是否公开模板
    created_by UUID, -- 创建者ID
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    status VARCHAR(20) DEFAULT 'active', -- 'active', 'inactive',
'deprecated'

    -- 索引
    CONSTRAINT uk_template_name_version UNIQUE (name, version),
    INDEX idx_template_category (category),
    INDEX idx_template_tags (tags),
    INDEX idx_template_status (status),
    INDEX idx_template_created_at (created_at)
);
```

2. API 文档实例表 (api_documents)

```
CREATE TABLE api_documents (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    template_id UUID REFERENCES openapi_templates(id) ON DELETE SET
```

```
NULL,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    version VARCHAR(50) NOT NULL,
    base_url VARCHAR(500) NOT NULL, -- 实际的基础URL
    server_url VARCHAR(500), -- 服务器URL（如果与base_url不同）

    -- 解析后的文档结构
    parsed_spec JSONB NOT NULL, -- 解析后的 OpenAPI 规范
    original_content TEXT NOT NULL, -- 原始文档内容
    content_type VARCHAR(20) NOT NULL, -- 'yaml' 或 'json'

    -- 配置信息
    auth_config JSONB, -- 认证配置
    rate_limit_config JSONB, -- 限流配置
    timeout_config JSONB, -- 超时配置

    -- 状态信息
    status VARCHAR(20) DEFAULT 'active', -- 'active', 'inactive',
'error'
    health_status VARCHAR(20) DEFAULT 'unknown', -- 'healthy',
'unhealthy', 'unknown'
    last_health_check TIMESTAMP WITH TIME ZONE,

    -- 统计信息
    total_endpoints INTEGER DEFAULT 0,
    active_endpoints INTEGER DEFAULT 0,

    -- 元数据
    created_by UUID,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

    -- 索引
    CONSTRAINT uk_api_name_version UNIQUE (name, version),
    INDEX idx_api_template_id (template_id),
    INDEX idx_api_status (status),
    INDEX idx_api_health_status (health_status),
    INDEX idx_api_created_at (created_at)
);
```

3. API 端点表 (api_endpoints)

```
CREATE TABLE api_endpoints (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    api_document_id UUID NOT NULL REFERENCES api_documents(id) ON DELETE
CASCADE,

    -- 端点信息
    path VARCHAR(500) NOT NULL, -- 路径，如 "/pets/{petId}"
```

```
    method VARCHAR(10) NOT NULL, -- HTTP 方法：GET, POST, PUT, DELETE,
PATCH
    operation_id VARCHAR(255), -- 操作ID
    summary TEXT,
    description TEXT,
    tags TEXT[], -- 标签数组

    -- 参数信息
    parameters JSONB, -- 参数定义
    request_body_schema JSONB, -- 请求体模式
    response_schemas JSONB, -- 响应模式

    -- 配置信息
    auth_required BOOLEAN DEFAULT false,
    rate_limit_enabled BOOLEAN DEFAULT false,
    timeout_ms INTEGER DEFAULT 30000, -- 超时时间（毫秒）

    -- 状态信息
    status VARCHAR(20) DEFAULT 'active', -- 'active', 'inactive',
'deprecated'
    is_deprecated BOOLEAN DEFAULT false,

    -- 统计信息
    call_count BIGINT DEFAULT 0, -- 调用次数
    success_count BIGINT DEFAULT 0, -- 成功次数
    error_count BIGINT DEFAULT 0, -- 错误次数
    avg_response_time_ms INTEGER, -- 平均响应时间

    -- 元数据
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

    -- 索引
    CONSTRAINT uk_endpoint_path_method UNIQUE (api_document_id, path,
method),
    INDEX idx_endpoint_api_document_id (api_document_id),
    INDEX idx_endpoint_method (method),
    INDEX idx_endpoint_status (status),
    INDEX idx_endpoint_tags (tags),
    INDEX idx_endpoint_operation_id (operation_id)
);
```

4. 资源引用表 (resource_references)

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

    -- 引用信息
    resource_type VARCHAR(50) NOT NULL, -- 'workflow_node',
'scheduled_task', 'form', 'dashboard'
```

```
    resource_id UUID NOT NULL, -- 引用资源的ID

    -- 被引用的API端点
    api_endpoint_id UUID NOT NULL REFERENCES api_endpoints(id) ON DELETE
CASCADE,

    -- 引用配置
    reference_config JSONB, -- 引用配置，如参数映射、认证信息等
    display_name VARCHAR(255), -- 显示名称
    description TEXT,

    -- 状态信息
    status VARCHAR(20) DEFAULT 'active', -- 'active', 'inactive',
'error'
    last_used_at TIMESTAMP WITH TIME ZONE,
    usage_count INTEGER DEFAULT 0,

    -- 元数据
    created_by UUID,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

    -- 索引
    INDEX idx_ref_resource_type_id (resource_type, resource_id),
    INDEX idx_ref_api_endpoint_id (api_endpoint_id),
    INDEX idx_ref_status (status),
    INDEX idx_ref_last_used (last_used_at)
);
```

5. API 调用日志表 (api_call_logs)

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

    -- 调用信息
    api_endpoint_id UUID NOT NULL REFERENCES api_endpoints(id) ON DELETE
CASCADE,
    resource_reference_id UUID REFERENCES resource_references(id) ON
DELETE SET NULL,

    -- 请求信息
    request_method VARCHAR(10) NOT NULL,
    request_url TEXT NOT NULL,
    request_headers JSONB,
    request_body TEXT,
    request_params JSONB,

    -- 响应信息
    response_status_code INTEGER,
    response_headers JSONB,
```

```
    response_body TEXT,

    -- 性能信息
    response_time_ms INTEGER,
    request_size_bytes INTEGER,
    response_size_bytes INTEGER,

    -- 错误信息
    error_message TEXT,
    error_type VARCHAR(50),

    -- 元数据
    client_ip INET,
    user_agent TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

    -- 索引
    INDEX idx_log_api_endpoint_id (api_endpoint_id),
    INDEX idx_log_resource_reference_id (resource_reference_id),
    INDEX idx_log_status_code (response_status_code),
    INDEX idx_log_created_at (created_at),
    INDEX idx_log_error_type (error_type)
);
```

6. API 健康检查表 (api_health_checks)

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

    -- 检查信息
    api_document_id UUID NOT NULL REFERENCES api_documents(id) ON DELETE
CASCADE,

    -- 检查结果
    check_type VARCHAR(50) NOT NULL, -- 'connectivity', 'endpoint',
'full'
    status VARCHAR(20) NOT NULL, -- 'success', 'failed', 'timeout'

    -- 详细信息
    response_time_ms INTEGER,
    error_message TEXT,
    details JSONB,

    -- 元数据
    checked_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

    -- 索引
    INDEX idx_health_api_document_id (api_document_id),
    INDEX idx_health_status (status),
```

```
    INDEX idx_health_checked_at (checked_at)
);
```

视图设计

1. API 端点资源视图 (api_endpoint_resources)

```
CREATE VIEW api_endpoint_resources AS
SELECT
    e.id as endpoint_id,
    e.path,
    e.method,
    e.operation_id,
    e.summary,
    e.description,
    e.tags,
    d.id as api_document_id,
    d.name as api_name,
    d.version as api_version,
    d.base_url,
    e.status as endpoint_status,
    d.status as api_status,
    e.call_count,
    e.success_count,
    e.error_count,
    e.avg_response_time_ms,
    COUNT(r.id) as reference_count
FROM api_endpoints e
JOIN api_documents d ON e.api_document_id = d.id
LEFT JOIN resource_references r ON e.id = r.api_endpoint_id AND r.status
= 'active'
GROUP BY e.id, d.id;
```

2. 资源引用统计视图 (resource_reference_stats)

```
CREATE VIEW resource_reference_stats AS
SELECT
    r.resource_type,
    r.resource_id,
    COUNT(r.id) as total_references,
    COUNT(CASE WHEN r.status = 'active' THEN 1 END) as
active_references,
    COUNT(CASE WHEN r.status = 'error' THEN 1 END) as error_references,
    SUM(r.usage_count) as total_usage,
    MAX(r.last_used_at) as last_used
FROM resource_references r
GROUP BY r.resource_type, r.resource_id;
```

索引优化

复合索引

```
-- 用于快速查找特定API文档的端点
CREATE INDEX idx_endpoints_doc_method_status ON
api_endpoints(api_document_id, method, status);

-- 用于资源引用查询
CREATE INDEX idx_refs_type_status_endpoint ON
resource_references(resource_type, status, api_endpoint_id);

-- 用于日志查询
CREATE INDEX idx_logs_endpoint_time ON api_call_logs(api_endpoint_id,
created_at DESC);
```

数据完整性约束

外键约束

```
-- 确保引用完整性
ALTER TABLE api_endpoints
ADD CONSTRAINT fk_endpoints_api_document
FOREIGN KEY (api_document_id) REFERENCES api_documents(id) ON DELETE
CASCADE;

ALTER TABLE resource_references
ADD CONSTRAINT fk_refs_endpoint
FOREIGN KEY (api_endpoint_id) REFERENCES api_endpoints(id) ON DELETE
CASCADE;
```

检查约束

```
-- 确保HTTP方法有效
ALTER TABLE api_endpoints
ADD CONSTRAINT chk_method
CHECK (method IN ('GET', 'POST', 'PUT', 'DELETE', 'PATCH', 'HEAD',
'OPTIONS'));

-- 确保状态值有效
ALTER TABLE api_documents
ADD CONSTRAINT chk_doc_status
CHECK (status IN ('active', 'inactive', 'error'));

ALTER TABLE api_endpoints
```

```
ADD CONSTRAINT chk_endpoint_status
CHECK (status IN ('active', 'inactive', 'deprecated'));
```

触发器

更新时间戳触发器

```
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = NOW();
    RETURN NEW;
END;
$$ language 'plpgsql';

-- 为相关表添加更新时间戳触发器
CREATE TRIGGER update_api_documents_updated_at
    BEFORE UPDATE ON api_documents
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_api_endpoints_updated_at
    BEFORE UPDATE ON api_endpoints
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_resource_references_updated_at
    BEFORE UPDATE ON resource_references
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();
```

统计更新触发器

```
-- 当API调用日志插入时，更新端点的统计信息
CREATE OR REPLACE FUNCTION update_endpoint_stats()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE api_endpoints
    SET
        call_count = call_count + 1,
        success_count = CASE WHEN NEW.response_status_code BETWEEN 200
AND 299
                             THEN success_count + 1 ELSE success_count
END,
        error_count = CASE WHEN NEW.response_status_code >= 400
                           THEN error_count + 1 ELSE error_count END,
        avg_response_time_ms = CASE
            WHEN avg_response_time_ms IS NULL THEN NEW.response_time_ms
            ELSE (avg_response_time_ms + NEW.response_time_ms) / 2
        END
    WHERE id = NEW.api_endpoint_id;
```

```
    RETURN NEW;
END;
$$ language 'plpgsql';

CREATE TRIGGER update_endpoint_stats_trigger
    AFTER INSERT ON api_call_logs
    FOR EACH ROW EXECUTE FUNCTION update_endpoint_stats();
```

使用场景

1. 前端展示

- 查询 `api_endpoint_resources` 视图获取所有可用的API端点
- 根据标签、状态等条件过滤
- 显示调用统计和健康状态

2. HTTP执行

- 从 `api_endpoints` 表获取端点配置
- 从 `api_documents` 表获取基础URL和认证配置
- 记录调用日志到 `api_call_logs` 表

3. 工作流节点引用

- 在 `resource_references` 表中创建引用记录
- `resource_type` 设为 'workflow_node'
- `resource_id` 为工作流节点ID
- 通过 `reference_config` 存储参数映射

4. 调度任务引用

- 在 `resource_references` 表中创建引用记录
- `resource_type` 设为 'scheduled_task'
- `resource_id` 为调度任务ID
- 通过 `reference_config` 存储调度配置

扩展建议

1. 分区表

对于 `api_call_logs` 表，建议按时间分区：

```
-- 按月分区
CREATE TABLE api_call_logs_2024_12 PARTITION OF api_call_logs
FOR VALUES FROM ('2024-12-01') TO ('2025-01-01');
```

2. 缓存表

创建热点数据缓存表：

```
CREATE TABLE api_endpoint_cache (
    endpoint_id UUID PRIMARY KEY,
    cache_data JSONB,
    cached_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    expires_at TIMESTAMP WITH TIME ZONE
);
```

3. 审计表

创建数据变更审计表：

```
CREATE TABLE api_audit_logs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    table_name VARCHAR(50) NOT NULL,
    record_id UUID NOT NULL,
    operation VARCHAR(20) NOT NULL, -- 'INSERT', 'UPDATE', 'DELETE'
    old_data JSONB,
    new_data JSONB,
    changed_by UUID,
    changed_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

这个数据库设计支持完整的API文档生命周期管理，从模板存储到实际执行，再到资源引用和监控。