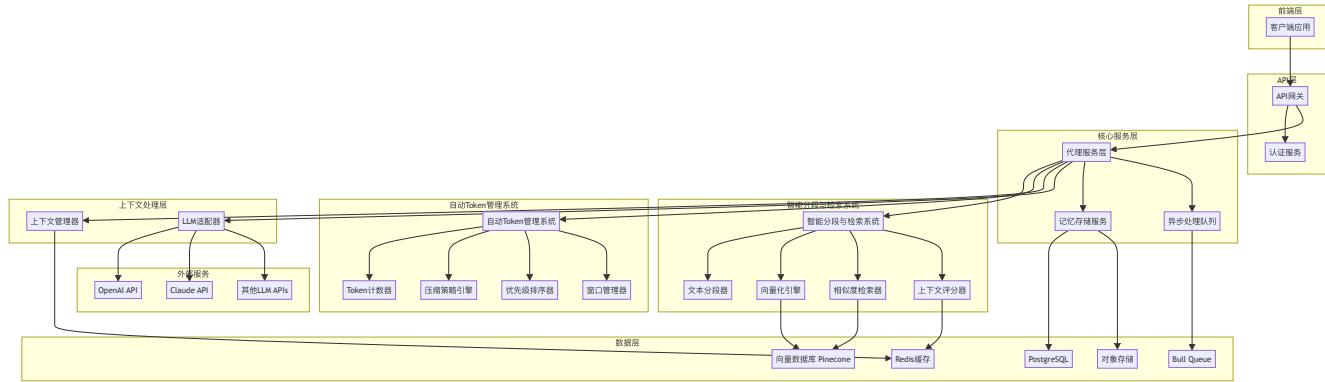


Supermemory 技术架构设计

1. 架构设计



2. 技术描述

- 前端:** React@18 + TypeScript + Tailwind CSS + Vite
- 后端:** Node.js + Express@4 + TypeScript
- 数据库:** PostgreSQL + Redis + Pinecone (向量数据库)
- 嵌入服务:** OpenAI text-embedding-3-small + sentence-transformers
- 消息队列:** Redis + Bull Queue
- 缓存系统:** Redis (多层缓存策略)
- 监控:** Prometheus + Grafana
- 部署:** Docker + Kubernetes

3. 路由定义

路由	用途
/dashboard	管理控制台主页，显示项目概览和使用统计
/projects	项目管理页面，创建和配置AI项目
/proxy-config	代理配置页面，设置LLM模型和记忆策略
/memory	记忆管理页面，查看和管理对话历史
/segmentation-config	智能分段配置页面，设置分段策略和检索参数

路由	用途
/token-management	Token管理页面，监控使用量和优化策略配置
/analytics	分析仪表板，性能和成本分析
/settings	系统设置页面，账户和安全配置
/api-docs	API文档页面，开发者集成指南

4. API定义

4.1 核心代理API

透明代理接口

POST /v1/chat/completions

请求参数:

参数名	参数类型	是否必需	描述
model	string	true	LLM模型名称 (gpt-4, claude-3, etc.)
messages	array	true	对话消息数组
temperature	number	false	生成温度 (0-2)
max_tokens	number	false	最大token数
stream	boolean	false	是否流式响应
memory_config	object	false	记忆配置参数

响应:

参数名	参数类型	描述
id	string	请求唯一标识
object	string	响应对象类型
created	number	创建时间戳
model	string	使用的模型
choices	array	生成的回复选项

参数名	参数类型	描述
usage	object	Token使用统计
memory_info	object	记忆处理信息

示例请求:

```
{
  "model": "gpt-4",
  "messages": [
    {"role": "user", "content": "你好，我想了解AI的发展历史"}
  ],
  "temperature": 0.7,
  "memory_config": {
    "enable_memory": true,
    "memory_depth": 10,
    "relevance_threshold": 0.8
  }
}
```

记忆管理API

```
GET /v1/memory/conversations
POST /v1/memory/clear
PUT /v1/memory/update
GET /v1/memory/search
POST /v1/memory/retrieve
```

智能分段配置API

```
GET /v1/segmentation/config
PUT /v1/segmentation/config
POST /v1/segmentation/test
GET /v1/segmentation/performance
```

Token管理API

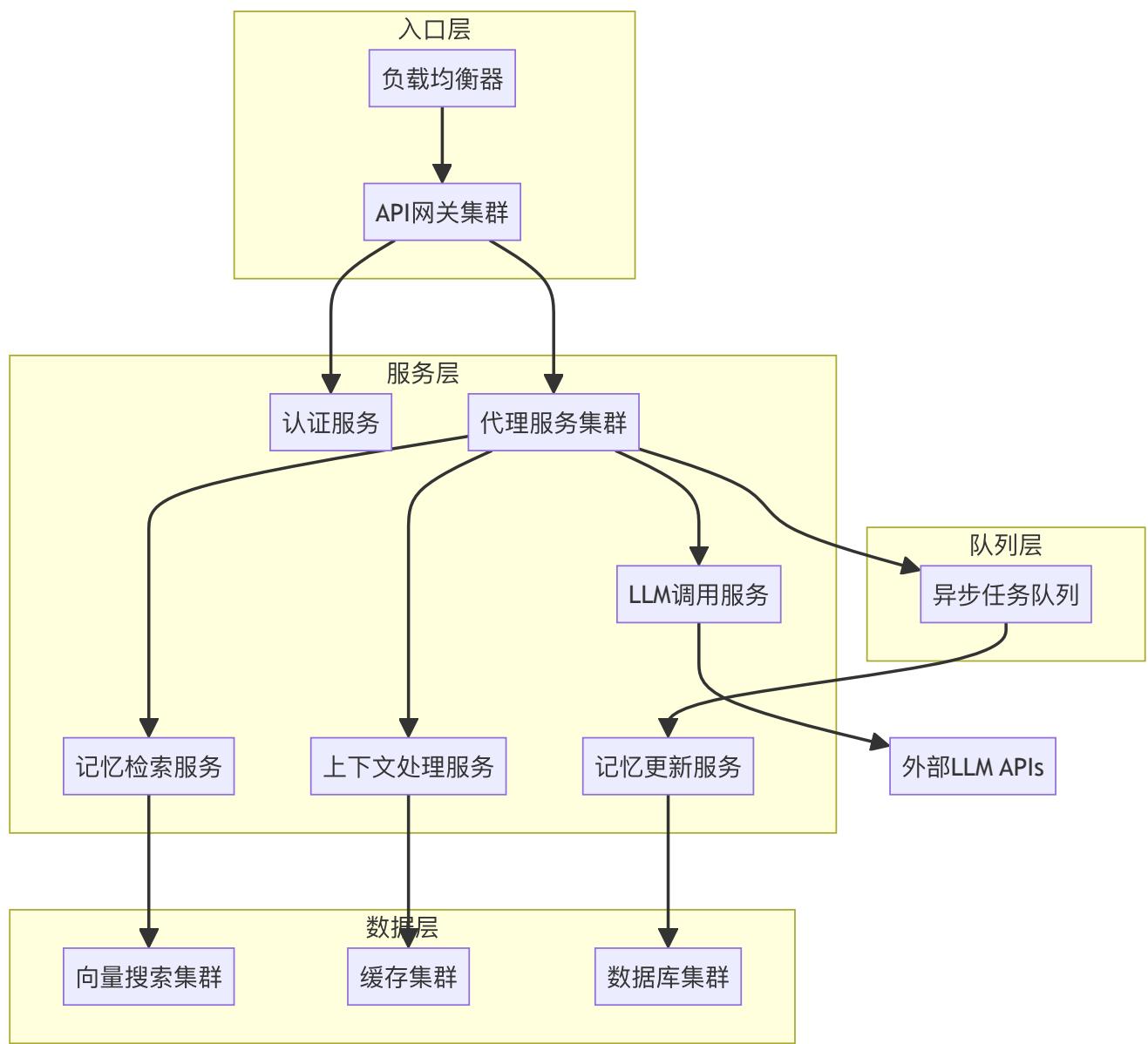
```
GET /v1/token/usage
GET /v1/token/statistics
PUT /v1/token/config
POST /v1/token/optimize
```

项目管理API

```
GET /v1/projects
POST /v1/projects
```

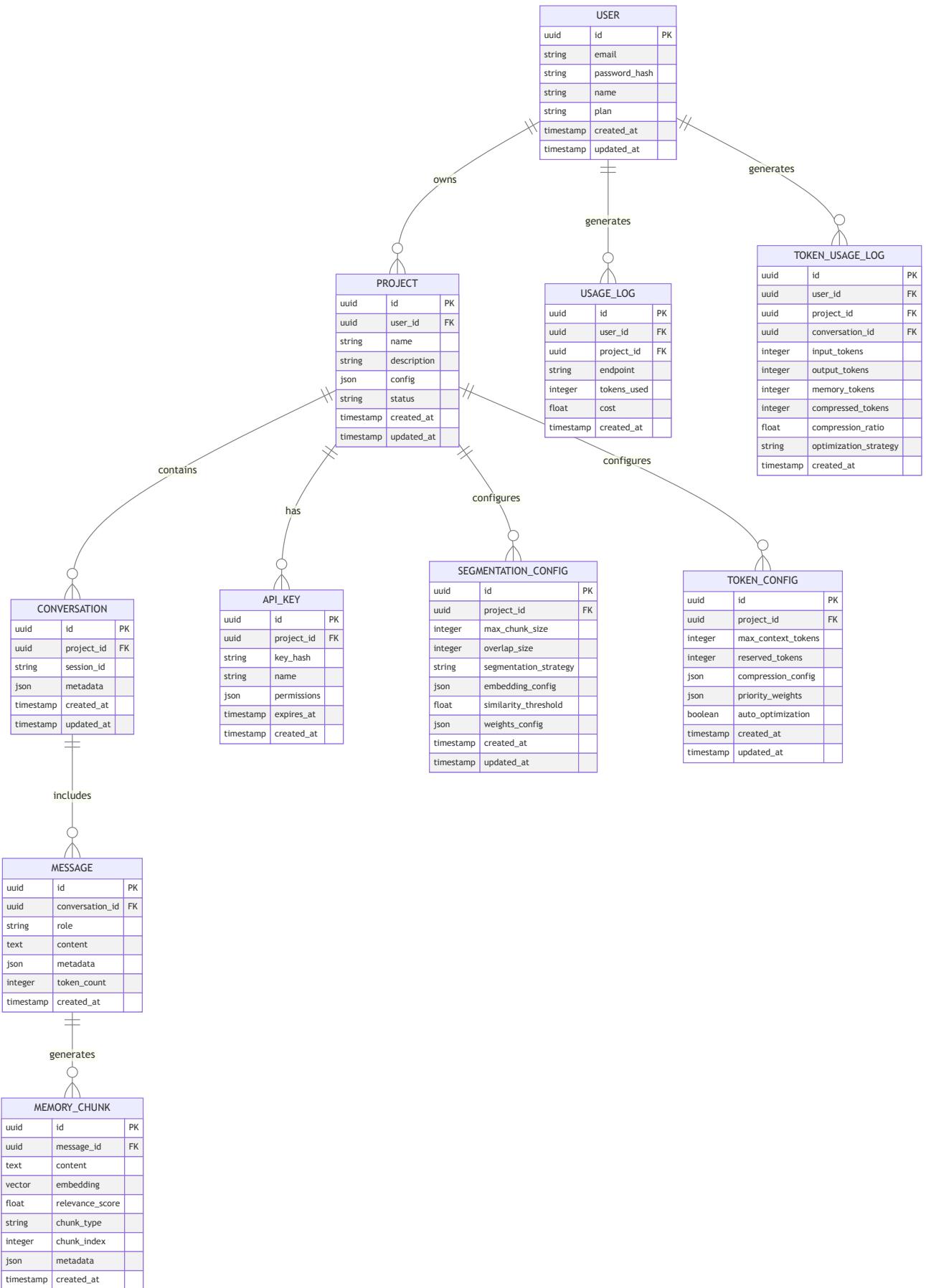
```
PUT /v1/projects/{id}  
DELETE /v1/projects/{id}
```

5. 服务器架构图



6. 数据模型

6.1 数据模型定义



6.2 数据定义语言

用户表 (users)

```
-- 创建用户表
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    name VARCHAR(100) NOT NULL,
    plan VARCHAR(20) DEFAULT 'free' CHECK (plan IN ('free', 'pro', 'premium')),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 创建索引
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_plan ON users(plan);
```

项目表 (projects)

```
-- 创建项目表
CREATE TABLE projects (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    config JSONB DEFAULT '{}',
    status VARCHAR(20) DEFAULT 'active' CHECK (status IN ('active', 'inactive', 'archived')),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 创建索引
CREATE INDEX idx_projects_user_id ON projects(user_id);
CREATE INDEX idx_projects_status ON projects(status);
```

对话表 (conversations)

```
-- 创建对话表
CREATE TABLE conversations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    project_id UUID NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
    session_id VARCHAR(255) NOT NULL,
    metadata JSONB DEFAULT '{}',
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 创建索引
CREATE INDEX idx_conversations_project_id ON conversations(project_id);
```

```
CREATE INDEX idx_conversations_session_id ON conversations(session_id)
CREATE INDEX idx_conversations_created_at ON conversations(created_at)
```

消息表 (messages)

```
-- 创建消息表
CREATE TABLE messages (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    conversation_id UUID NOT NULL REFERENCES conversations(id) ON DELETE CASCADE,
    role VARCHAR(20) NOT NULL CHECK (role IN ('user', 'assistant', 'assistant_v2')),
    content TEXT NOT NULL,
    metadata JSONB DEFAULT '{}',
    token_count INTEGER DEFAULT 0,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 创建索引
CREATE INDEX idx_messages_conversation_id ON messages(conversation_id);
CREATE INDEX idx_messages_created_at ON messages(created_at DESC);
CREATE INDEX idx_messages_token_count ON messages(token_count);
```

记忆块表 (memory_chunks)

```
-- 创建记忆块表
CREATE TABLE memory_chunks (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    message_id UUID NOT NULL REFERENCES messages(id) ON DELETE CASCADE,
    content TEXT NOT NULL,
    embedding VECTOR(1536), -- 使用pgvector扩展
    relevance_score FLOAT DEFAULT 0.0,
    chunk_type VARCHAR(50) DEFAULT 'semantic' CHECK (chunk_type IN ('semantic', 'code')),
    chunk_index INTEGER DEFAULT 0,
    metadata JSONB DEFAULT '{}',
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 创建向量索引
CREATE INDEX idx_memory_chunks_embedding ON memory_chunks USING ivff_sq8;
CREATE INDEX idx_memory_chunks_message_id ON memory_chunks(message_id);
CREATE INDEX idx_memory_chunks_relevance_score ON memory_chunks(relevance_score);
CREATE INDEX idx_memory_chunks_chunk_type ON memory_chunks(chunk_type);
CREATE INDEX idx_memory_chunks_chunk_index ON memory_chunks(chunk_index);
```

分段配置表 (segmentation_configs)

```
-- 创建分段配置表
CREATE TABLE segmentation_configs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    project_id UUID NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
```

```
max_chunk_size INTEGER DEFAULT 512,
overlap_size INTEGER DEFAULT 50,
segmentation_strategy VARCHAR(50) DEFAULT 'semantic' CHECK (segm
embedding_config JSONB DEFAULT '{"model": "text-embedding-3-smal
similarity_threshold FLOAT DEFAULT 0.7,
weights_config JSONB DEFAULT '{"semantic": 0.4, "temporal": 0.3,
created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

-- 创建索引

```
CREATE INDEX idx_segmentation_configs_project_id ON segmentation_con
CREATE UNIQUE INDEX idx_segmentation_configs_project_unique ON segme
```

Token配置表 (token_configs)

-- 创建Token配置表

```
CREATE TABLE token_configs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    project_id UUID NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
    max_context_tokens INTEGER DEFAULT 4096,
    reserved_tokens INTEGER DEFAULT 1000,
    compression_config JSONB DEFAULT '{"enable_compression": true, "priorit
    priority_weights JSONB DEFAULT '{"temporal": 0.3, "relevance": 0.7},
    auto_optimization BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

-- 创建索引

```
CREATE INDEX idx_token_configs_project_id ON token_configs(project_i
CREATE UNIQUE INDEX idx_token_configs_project_unique ON token_config
```

Token使用日志表 (token_usage_logs)

-- 创建Token使用日志表

```
CREATE TABLE token_usage_logs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    project_id UUID REFERENCES projects(id) ON DELETE SET NULL,
    conversation_id UUID REFERENCES conversations(id) ON DELETE SET NULL,
    input_tokens INTEGER DEFAULT 0,
    output_tokens INTEGER DEFAULT 0,
    memory_tokens INTEGER DEFAULT 0,
    compressed_tokens INTEGER DEFAULT 0,
    compression_ratio FLOAT DEFAULT 1.0,
    optimization_strategy VARCHAR(100),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

```
-- 创建索引
CREATE INDEX idx_token_usage_logs_user_id ON token_usage_logs(user_i
CREATE INDEX idx_token_usage_logs_project_id ON token_usage_logs(pro
CREATE INDEX idx_token_usage_logs_conversation_id ON token_usage_log
CREATE INDEX idx_token_usage_logs_created_at ON token_usage_logs(cre

-- 创建分区表 (按月分区)
CREATE TABLE token_usage_logs_y2024m01 PARTITION OF token_usage_logs
FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');
```

API密钥表 (api_keys)

```
-- 创建API密钥表
CREATE TABLE api_keys (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    project_id UUID NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
    key_hash VARCHAR(255) NOT NULL UNIQUE,
    name VARCHAR(100) NOT NULL,
    permissions JSONB DEFAULT '{}',
    expires_at TIMESTAMP WITH TIME ZONE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 创建索引
CREATE INDEX idx_api_keys_project_id ON api_keys(project_id);
CREATE INDEX idx_api_keys_key_hash ON api_keys(key_hash);
```

使用日志表 (usage_logs)

```
-- 创建使用日志表
CREATE TABLE usage_logs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    project_id UUID REFERENCES projects(id) ON DELETE SET NULL,
    endpoint VARCHAR(255) NOT NULL,
    tokens_used INTEGER DEFAULT 0,
    cost DECIMAL(10,6) DEFAULT 0.0,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 创建索引
CREATE INDEX idx_usage_logs_user_id ON usage_logs(user_id);
CREATE INDEX idx_usage_logs_project_id ON usage_logs(project_id);
CREATE INDEX idx_usage_logs_created_at ON usage_logs(created_at DESC)

-- 创建分区表 (按月分区)
CREATE TABLE usage_logs_y2024m01 PARTITION OF usage_logs
FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');
```

初始化数据

```
-- 插入示例用户
INSERT INTO users (email, password_hash, name, plan) VALUES
('admin@supermemory.ai', '$2b$10$example_hash', 'Admin User', 'enter'
('demo@example.com', '$2b$10$example_hash', 'Demo User', 'pro');

-- 插入示例项目
INSERT INTO projects (user_id, name, description, config) VALUES
((SELECT id FROM users WHERE email = 'demo@example.com'),
'Demo Chat Bot',
'A demonstration chatbot with memory capabilities',
'{"model": "gpt-4", "memory_depth": 10, "temperature": 0.7}');

-- 插入默认分段配置
INSERT INTO segmentation_configs (project_id, max_chunk_size, overlap)
((SELECT id FROM projects WHERE name = 'Demo Chat Bot'),
512, 50, 'semantic', 0.7);

-- 插入默认Token配置
INSERT INTO token_configs (project_id, max_context_tokens, reserved_percent)
((SELECT id FROM projects WHERE name = 'Demo Chat Bot'),
4096, 1000, true);

-- 插入示例API密钥
INSERT INTO api_keys (project_id, key_hash, name, permissions) VALUE
((SELECT id FROM projects WHERE name = 'Demo Chat Bot'),
'$2b$10$demo_api_key_hash',
'Demo API Key',
'{"read": true, "write": true, "admin": false}');
```