VALORANT 智能战术助手 数据库设计说明书

1. 数据库概念设计

1.1 ER 图（实体关系图）

关系说明

User → ChatHistory: 一对多关系（1:N），一个用户可以有多条聊天记录

User → Tactic: 一对多关系（1:N），一个用户可以创建多个战术

User → FavoriteTactic: 一对多关系（1:N），一个用户可以收藏多个战术

Tactic → DrawingElement: 一对多关系（1:N），一个战术可以包含多个绘制元素

Tactic → Map: 多对一关系（N:1），多个战术可以针对同一地图

Agent → Ability: 一对多关系（1:N），一个英雄可以有多个技能

FavoriteTactic: 联接表，实现用户与战术的多对多关系（M:N）

1.2 实体说明

实体	说明	主要属性
User	用户	id, username, email, password_hash

实体	说明	主要属性
ChatHistory	聊天历史	id, user_id, role, content, timestamp
Agent	英雄	id, name, role, description, image_url
Ability	技能	id, agent_id, key, name, description
Map	地图	id, name, description, sites, key_points
Tactic	战术	id, name, map_id, description, elements
DrawingElement	绘制元素	id, tactic_id, type, position, color
FavoriteTactic	收藏战术	id, user_id, tactic_id, created_at

1.3 关系数据模型（表格）

关系（表）	主键（PK）	外键（FK）	说明
User	id	-	存储系统用户的账号与基础信息
ChatHistory	id	user_id → User.id	记录用户与AI的每条对话消息
Agent	id	-	存储游戏英雄的基础信息

关系（表）	主键（PK）	外键（FK）	说明
Ability	id	agent_id → Agent.id	存储每个英雄对应的技能
Map	id	-	存储地图及其站点、关键点等信息
Tactic	id	user_id → User.id；map_id → Map.id	存储用户或系统预设的战术方案
DrawingElement	id	tactic_id → Tactic.id	存储战术板上的具体绘制元素（点、线、区域等）
FavoriteTactic	id	user_id → User.id；tactic_id → Tactic.id	关联用户与其收藏的战术，实现多对多关系

2. 数据库逻辑设计

2.1 表结构设计

2.1.1 英雄表 (agents)

```
CREATE TABLE agents (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(50) NOT NULL UNIQUE,
    role VARCHAR(20) NOT NULL,    -- 'Duelist', 'Sentinel', 'Initiator', 'Controller'
    description TEXT,
    image_url VARCHAR(255),
    nationality VARCHAR(50),
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);


CREATE INDEX idx_agents_name ON agents(name);
CREATE INDEX idx_agents_role ON agents(role);
```

2.1.2 技能表 (abilities)

```
CREATE TABLE abilities (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    agent_id UUID NOT NULL REFERENCES agents(id) ON DELETE CASCADE,
    key VARCHAR(10) NOT NULL,   -- 'E', 'Q', 'C', 'X'
    name VARCHAR(50) NOT NULL,
    description TEXT NOT NULL,
    cooldown INT,   -- 秒
    cost INT,   -- 经济成本
    type VARCHAR(20),   -- 'ability' or 'ultimate'
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);


CREATE INDEX idx_abilities_agent_id ON abilities(agent_id);
CREATE INDEX idx_abilities_key ON abilities(key);
```

2.1.3 地图表 (maps)

```
CREATE TABLE maps (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(50) NOT NULL UNIQUE,
    description TEXT,
    sites VARCHAR(20),   -- 'A,B' or 'A,B,C'
    key_points TEXT,   -- JSON 格式
    image_url VARCHAR(255),
    difficulty_level INT,   -- 1-5
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);


CREATE INDEX idx_maps_name ON maps(name);
```

2.1.4 战术表 (tactics)

```
CREATE TABLE tactics (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    map_id UUID NOT NULL REFERENCES maps(id),
    name VARCHAR(100) NOT NULL,
```

```
    description TEXT,
    elements JSONB,   -- 绘制元素 JSON
    timeline JSONB,   -- 时间轴 JSON
    is_preset BOOLEAN DEFAULT FALSE,   -- 是否为预设战术
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    view_count INT DEFAULT 0
);

CREATE INDEX idx_tactics_user_id ON tactics(user_id);
CREATE INDEX idx_tactics_map_id ON tactics(map_id);
CREATE INDEX idx_tactics_is_preset ON tactics(is_preset);
```

3. 对象关系映射 (ORM)

3.1 Python SQLAlchemy 映射

```
from sqlalchemy import Column, String, Text, Integer, Boolean, DateTime, ForeignKey, JSONB
from sqlalchemy.dialects.postgresql import UUID
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship
from uuid import uuid4
from datetime import datetime

Base = declarative_base()

class Agent(Base):
    __tablename__ = "agents"
    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    name = Column(String(50), unique=True, nullable=False, index=True)
    role = Column(String(20), nullable=False)
    description = Column(Text)
    image_url = Column(String(255))
    nationality = Column(String(50))
    created_at = Column(DateTime, default=datetime.utcnow)
    updated_at = Column(DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)
    abilities = relationship("Ability", back_populates="agent", cascade="all, delete-orphan")

class Ability(Base):
    __tablename__ = "abilities"
    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    agent_id = Column(UUID(as_uuid=True), ForeignKey("agents.id", ondelete="CASCADE"),
nullable=False)
    key = Column(String(10), nullable=False)
    name = Column(String(50), nullable=False)
    description = Column(Text, nullable=False)
```

```
    cooldown = Column(Integer)
    cost = Column(Integer)
    type = Column(String(20))
    created_at = Column(DateTime, default=datetime.utcnow)
    agent = relationship("Agent", back_populates="abilities")

class Map(Base):
    __tablename__ = "maps"
    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    name = Column(String(50), unique=True, nullable=False, index=True)
    description = Column(Text)
    sites = Column(String(20))
    key_points = Column(String)
    image_url = Column(String(255))
    difficulty_level = Column(Integer)
    created_at = Column(DateTime, default=datetime.utcnow)
    updated_at = Column(DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)
    tactics = relationship("Tactic", back_populates="map")

class Tactic(Base):
    __tablename__ = "tactics"
    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    user_id = Column(UUID(as_uuid=True), ForeignKey("users.id", ondelete="CASCADE"),
nullable=False)
    map_id = Column(UUID(as_uuid=True), ForeignKey("maps.id"), nullable=False)
    name = Column(String(100), nullable=False)
    description = Column(Text)
    elements = Column(JSONB)
    timeline = Column(JSONB)
    is_preset = Column(Boolean, default=False)
    view_count = Column(Integer, default=0)
    created_at = Column(DateTime, default=datetime.utcnow)
    updated_at = Column(DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)
    user = relationship("User", back_populates="tactics")
    map = relationship("Map", back_populates="tactics")
```

3.2 C# 实体类映射

```
csharp
public class Agent
{
    public Guid Id { get; set; }
    public string Name { get; set; }
```

```
        public string Role { get; set; }
        public string Description { get; set; }
        public string ImageUrl { get; set; }
        public virtual ICollection<Ability> Abilities { get; set; }
}

public class Ability
{
        public Guid Id { get; set; }
        public Guid AgentId { get; set; }
        public string Key { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public int? Cooldown { get; set; }
        public int? Cost { get; set; }
        public virtual Agent Agent { get; set; }
}

public class Map
{
        public Guid Id { get; set; }
    <br/><br/>
        public string Name { get; set; }
        public string Description { get; set; }
        public string Sites { get; set; }
        public string ImageUrl { get; set; }
        public virtual ICollection<Tactic> Tactics { get; set; }
}

public class Tactic
{
        public Guid Id { get; set; }
        public Guid MapId { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public string Elements { get; set; }
        public string Timeline { get; set; }
        public virtual Map Map { get; set; }
}
```

4. 关系数据模型体现总结

4.1 关系数据模型在本系统中的体现

主键与实体完整性

每个表都有唯一的主键（id），确保每条记录的唯一性

使用 UUID 类型，保证全局唯一性和分布式系统兼容性
体现位置：所有表的 id 字段
外键与参照完整性
通过外键建立表之间的关系，维护数据一致性
使用 ON DELETE CASCADE 确保删除父记录时自动清理子记录
体现在 tactics.map_id → maps.id、abilities.agent_id → agents.id 等关系中
唯一约束
agents(name) 和 maps(name) 字段唯一，防止重复
确保数据的唯一性和完整性
关系类型
一对多 (1:N)：Agent → Ability、Map → Tactic
多对一 (N:1)：Ability → Agent、Tactic → Map
范式设计
采用第三范式 (3NF) 设计
复杂数据使用 JSONB 存储（elements、timeline）
不存储可以通过外键推导的数据