微信小程序数据库需求文档

一、前后端对接规范

1. 通用响应格式

```
{
  "code": 200,          // 状态码：200成功，400请求错误，401未授权，500服务器错误
  "message": "success", // 状态描述
  "data": {             // 具体数据
    // 具体字段
  }
}
```

2. 分页参数规范

```
{
  "page": 1,         // 当前页码，从1开始
  "pageSize": 10,    // 每页条数
  "total": 100,      // 总条数
  "items": []        // 具体数据列表
}
```

3. 时间格式

所有时间字段统一使用 ISO 8601 格式：`YYYY-MM-DDTHH:mm:ss.sssZ`

4. 文件上传

- 使用 multipart/form-data 格式
- 视频文件支持格式：mp4, mov
- 图片文件支持格式：jpg, png, gif
- 单个文件大小限制：视频50MB，图片5MB

二、数据库表设计及接口定义

1. 用户表 (users)

需求来源：

- 用户信息展示：`miniprogram/pages/profile/profile.wxml`
- 用户关注/粉丝统计：`miniprogram/pages/discovery/discovery.ts` (第2-193行)

字段设计：

- id: string (主键)
- openid: string (微信用户唯一标识)
- nickname: string (用户昵称)
- avatar_url: string (头像URL)

- created_at: timestamp (创建时间)
- updated_at: timestamp (更新时间)
- following_count: integer (关注数)
- followers_count: integer (粉丝数)
- likes_count: integer (获赞数)

数据插入位置：

- 用户基本信息：`miniprogram/pages/video-detail/video-detail.ts` 第17-19行 (用户名、头像等)
- 关注状态：`miniprogram/pages/video-detail/video-detail.ts` 第14行 (isFollowing)
- 点赞状态：`miniprogram/pages/video-detail/video-detail.ts` 第15行 (hasLiked)

2. 视频表 (videos)

需求来源：

- 视频列表展示：`miniprogram/pages/discovery/discovery.wxml` (第2-37行)
- 视频详情页：`miniprogram/pages/video-detail/video-detail.ts` (第2-77行)
- 视频上传：`miniprogram/pages/upload/upload.ts` (第2-55行)

字段设计：

- id: string (主键)
- user_id: string (外键，关联用户表)
- title: string (视频标题，参考 video-detail.ts 第49行)
- description: text (视频描述，参考 video-detail.ts 第50行)
- video_url: string (视频文件URL)
- thumbnail_url: string (缩略图URL，参考 discovery.ts 第15行)
- category_id: string (外键，关联分类表)
- visibility: enum ('public', 'friends', 'private') (可见性，参考 upload.ts 第3行)
- views_count: integer (观看次数)
- likes_count: integer (点赞数)
- comments_count: integer (评论数)
- created_at: timestamp (创建时间)
- updated_at: timestamp (更新时间)

数据插入位置：

- 视频基本信息：`miniprogram/pages/video-detail/video-detail.ts` 第49-58行
- 视频列表数据：`miniprogram/pages/discovery/discovery.ts` 第15-93行 (videoRows数组)
- 视频分类：`miniprogram/pages/discovery/discovery.ts` 第6-14行 (tabs数组)

3. 分类表 (categories)

需求来源：

- 分类标签：`miniprogram/pages/discovery/discovery.ts` (第6-14行)

字段设计：

- id: string (主键)
- name: string (分类名称，如：体操区、舞蹈区、运动区等)
- description: string (分类描述)
- sort_order: integer (排序顺序)
- created_at: timestamp (创建时间)
- updated_at: timestamp (更新时间)

数据插入位置：

- 分类列表：`miniprogram/pages/discovery/discovery.wxml` 第19-21行 (tabs遍历)

4. 评论表 (comments)

需求来源：

- 评论列表：`miniprogram/pages/video-detail/video-detail.ts` (第17-39行)

字段设计：

- id: string (主键)
- video_id: string (外键，关联视频表)
- user_id: string (外键，关联用户表)
- content: text (评论内容)
- likes_count: integer (点赞数)
- reply_count: integer (回复数)
- parent_id: string (父评论ID，用于回复功能)
- created_at: timestamp (创建时间)
- updated_at: timestamp (更新时间)

数据插入位置：

- 评论数据：`miniprogram/pages/video-detail/video-detail.ts` 第17-39行 (commentList数组)
- 评论计数：`miniprogram/pages/video-detail/video-detail.ts` 第11行 (comments)

5. 点赞表 (likes)

需求来源：

- 视频点赞：`miniprogram/pages/video-detail/video-detail.ts` (第13行)
- 评论点赞：`miniprogram/pages/video-detail/video-detail.ts` (第27行)

字段设计：

- id: string (主键)
- user_id: string (外键，关联用户表)
- target_id: string (被点赞对象ID)
- target_type: enum ('video', 'comment') (点赞类型)
- created_at: timestamp (创建时间)

数据插入位置：

- 视频点赞状态：`miniprogram/pages/video-detail/video-detail.ts` 第15行 (hasLiked)
- 评论点赞数：`miniprogram/pages/video-detail/video-detail.ts` 第23行 (likes)

6. 关注关系表 (follows)

需求来源：

- 关注功能：`miniprogram/pages/discovery/discovery.ts` (第7行)

字段设计：

- id: string (主键)
- follower_id: string (关注者ID，外键关联用户表)
- following_id: string (被关注者ID，外键关联用户表)
- created_at: timestamp (创建时间)

数据插入位置：

- 关注状态：`miniprogram/pages/video-detail/video-detail.ts` 第14行 (isFollowing)
- 关注列表：`miniprogram/pages/discovery/discovery.ts` 第15-25行 (关注分类下的视频)

7. 用户学习记录表 (learning_records)

需求来源：

- 学习页面：`miniprogram/pages/learning/learning.ts` (第2-38行)
- 学习进度：`miniprogram/pages/learning/learning.wxml` (第2-25行)

字段设计：

- id: string (主键)
- user_id: string (外键，关联用户表)
- video_id: string (外键，关联视频表)
- progress: integer (学习进度，单位：秒)
- is_completed: boolean (是否完成)
- last_watched_at: timestamp (最后观看时间)
- created_at: timestamp (创建时间)
- updated_at: timestamp (更新时间)

数据插入位置：

- 学习列表：`miniprogram/pages/learning/learning.ts` 第19-38行 (wantToLearnList)
- 学习进度：需要在视频播放组件中添加进度记录功能

8. 搜索历史表 (search_history)

需求来源：

- 搜索功能：`miniprogram/pages/discovery/discovery.ts` (第39-60行)
- 搜索框：`miniprogram/pages/discovery/discovery.wxml` (第10-14行)

字段设计：

- id: string (主键)
- user_id: string (外键，关联用户表)
- keyword: string (搜索关键词)
- created_at: timestamp (创建时间)

数据插入位置：

- 搜索记录：`miniprogram/pages/discovery/discovery.ts` 第39-60行 (onSearch函数)

数据库索引建议

1. users表：
   - openid (唯一索引)
   - nickname (普通索引)
2. videos表：
   - user_id (普通索引)
   - category_id (普通索引)
   - created_at (普通索引)
   - views_count (普通索引)
3. comments表：
   - video_id (普通索引)
   - user_id (普通索引)
   - parent_id (普通索引)
4. likes表：
   - user_id, target_id, target_type (联合唯一索引)
5. follows表：
   - follower_id, following_id (联合唯一索引)
6. learning_records表：
   - user_id, video_id (联合唯一索引)
   - last_watched_at (普通索引)

注意事项

1. 所有表都应该使用软删除机制，添加 deleted_at 字段
2. 需要添加适当的数据库约束来保证数据完整性
3. 对于大型文本内容，考虑使用文本搜索引擎（如Elasticsearch）来提升搜索性能

4. 考虑使用缓存机制（如Redis）来缓存热门视频、用户信息等数据
5. 重要的统计数据（如点赞数、评论数）考虑使用计数器服务来处理
6. 文件存储（视频、图片等）建议使用对象存储服务

API 接口路径建议

1. 用户相关：
   - GET /api/users/:id - 获取用户信息
   - PUT /api/users/:id - 更新用户信息
   - GET /api/users/:id/followers - 获取用户粉丝列表
   - GET /api/users/:id/following - 获取用户关注列表
2. 视频相关：
   - GET /api/videos - 获取视频列表
   - POST /api/videos - 上传视频
   - GET /api/videos/:id - 获取视频详情
   - PUT /api/videos/:id - 更新视频信息
   - DELETE /api/videos/:id - 删除视频
   - POST /api/videos/:id/like - 点赞视频
   - POST /api/videos/:id/unlike - 取消点赞
3. 评论相关：
   - GET /api/videos/:id/comments - 获取视频评论列表
   - POST /api/videos/:id/comments - 发表评论
   - DELETE /api/comments/:id - 删除评论
   - POST /api/comments/:id/like - 点赞评论
   - POST /api/comments/:id/unlike - 取消点赞评论
4. 学习记录相关：
   - GET /api/learning-records - 获取学习记录
   - POST /api/learning-records - 创建/更新学习记录
   - GET /api/learning-records/:video_id/progress - 获取特定视频的学习进度

三、接口详细定义

1. 用户相关接口

1.1 获取用户信息

```
GET /api/users/:id

// 响应示例
{
  "code": 200,
  "message": "success",
  "data": {
    "id": "string",
```

```
    "nickname": "string",
    "avatar_url": "string",
    "following_count": 0,
    "followers_count": 0,
    "likes_count": 0
  }
}

// 调用示例 (miniprogram/pages/video-detail/video-detail.ts 第17-19行)
wx.request({
  url: `${BASE_URL}/api/users/${userId}`,
  method: 'GET',
  success: (res) => {
    if (res.data.code === 200) {
      this.setData({
        'video.author': res.data.data.nickname
      });
    }
  }
});
```

1.2 更新用户信息

```
PUT /api/users/:id
Content-Type: application/json

// 请求体
{
  "nickname": "string",
  "avatar_url": "string"
}

// 响应示例
{
  "code": 200,
  "message": "success",
  "data": null
}
```

2. 视频相关接口

2.1 获取视频列表

```
GET /api/videos
参数:
- page: number       // 页码
- pageSize: number   // 每页条数
- category_id: string // 分类ID
- user_id: string    // 用户ID，获取用户上传的视频

// 响应示例
{
  "code": 200,
  "message": "success",
```

```
  "data": {
    "page": 1,
    "pageSize": 10,
    "total": 100,
    "items": [{
      "id": "string",
      "title": "string",
      "thumbnail_url": "string",
      "author": {
        "id": "string",
        "nickname": "string",
        "avatar_url": "string"
      },
      "views_count": 0,
      "likes_count": 0,
      "comments_count": 0
    }]
  }
}

// 调用示例 (miniprogram/pages/discovery/discovery.ts 第15-93行)
wx.request({
  url: `${BASE_URL}/api/videos`,
  method: 'GET',
  data: {
    page: 1,
    pageSize: 10,
    category_id: this.data.activeTab
  },
  success: (res) => {
    if (res.data.code === 200) {
      this.setData({
        videoRows: this.formatVideoRows(res.data.data.items)
      });
    }
  }
});
```

2.2 上传视频

```
POST /api/videos
Content-Type: multipart/form-data

// 表单字段
- title: string          // 视频标题
- description: string    // 视频描述
- category_id: string    // 分类ID
- visibility: string     // 可见性
- video_file: File       // 视频文件
- thumbnail_file: File   // 缩略图文件

// 响应示例
{
  "code": 200,
  "message": "success",
```

```
  "data": {
    "id": "string",
    "video_url": "string",
    "thumbnail_url": "string"
  }
}

// 调用示例 (miniprogram/pages/upload/upload.ts 第39-60行)
wx.uploadFile({
  url: `${BASE_URL}/api/videos`,
  filePath: tempFilePath,
  name: 'video_file',
  formData: {
    title: this.data.title,
    description: this.data.description,
    category_id: this.data.categoryId,
    visibility: this.data.visibility
  },
  success: (res) => {
    const data = JSON.parse(res.data);
    if (data.code === 200) {
      wx.showToast({ title: '上传成功' });
    }
  }
});
```

四、错误码定义

1. 通用错误码

- 200: 成功
- 400: 请求参数错误
- 401: 未授权
- 403: 权限不足
- 404: 资源不存在
- 500: 服务器错误

2. 业务错误码

- 1001: 用户未登录
- 1002: 用户不存在
- 1003: 视频不存在
- 1004: 分类不存在
- 1005: 评论不存在
- 1006: 文件上传失败
- 1007: 文件格式不支持
- 1008: 文件大小超限

五、开发注意事项

1. 安全性

- 所有接口必须通过 HTTPS 访问
- 需要登录的接口在 Header 中携带 token
- 文件上传前先获取临时上传凭证

2. 异常处理

- 网络请求超时设置为15秒
- 请求失败后最多重试3次
- 上传大文件支持断点续传
- 统一的错误提示处理

数据库插入位置详细说明（MySQL）

1. 用户表 (users)

```
CREATE TABLE users (
    id VARCHAR(32) PRIMARY KEY,
    openid VARCHAR(32) UNIQUE NOT NULL,
    nickname VARCHAR(50) NOT NULL,
    avatar_url VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    following_count INT DEFAULT 0,
    followers_count INT DEFAULT 0,
    likes_count INT DEFAULT 0,
    deleted_at TIMESTAMP NULL,
    INDEX idx_nickname (nickname),
    INDEX idx_openid (openid)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

数据插入位置及SQL示例：

1. 用户基本信息（`miniprogram/pages/video-detail/video-detail.ts` 第17-19行）

```
-- 查询用户信息
SELECT nickname, avatar_url FROM users WHERE id = ? AND deleted_at IS NULL;

-- 前端展示代码
this.setData({
    'commentList[0].username': result.nickname
});
```

2. 关注状态（`miniprogram/pages/video-detail/video-detail.ts` 第14行）

```sql
-- 查询是否关注
SELECT COUNT(*) as is_following
FROM follows
WHERE follower_id = ? AND following_id = ? AND deleted_at IS NULL;

-- 前端展示代码
this.setData({
    isFollowing: result.is_following > 0
});
```

2. 视频表 (videos)

```sql
CREATE TABLE videos (
    id VARCHAR(32) PRIMARY KEY,
    user_id VARCHAR(32) NOT NULL,
    title VARCHAR(100) NOT NULL,
    description TEXT,
    video_url VARCHAR(255) NOT NULL,
    thumbnail_url VARCHAR(255),
    category_id VARCHAR(32) NOT NULL,
    visibility ENUM('public', 'friends', 'private') DEFAULT 'public',
    views_count INT DEFAULT 0,
    likes_count INT DEFAULT 0,
    comments_count INT DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP NULL,
    INDEX idx_user_id (user_id),
    INDEX idx_category_id (category_id),
    INDEX idx_created_at (created_at),
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (category_id) REFERENCES categories(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

数据插入位置及SQL示例：

1. 视频列表数据 (`miniprogram/pages/discovery/discovery.ts` 第15-93行)

```sql
-- 查询视频列表
SELECT
    v.id, v.title, v.thumbnail_url,
    u.nickname as author_name,
    u.avatar_url as author_avatar,
    v.views_count, v.likes_count, v.comments_count
FROM videos v
JOIN users u ON v.user_id = u.id
WHERE v.category_id = ?
    AND v.visibility = 'public'
    AND v.deleted_at IS NULL
ORDER BY v.created_at DESC
LIMIT ?, ?;

-- 前端展示代码
this.setData({
```

```
    videoRows: this.formatVideoRows(results)
});
```

2. 视频详情 (`miniprogram/pages/video-detail/video-detail.ts` 第49-58行)

```
-- 查询视频详情
SELECT
    v.*,
    u.nickname as author_name,
    u.avatar_url as author_avatar
FROM videos v
JOIN users u ON v.user_id = u.id
WHERE v.id = ? AND v.deleted_at IS NULL;

-- 更新观看次数
UPDATE videos
SET views_count = views_count + 1
WHERE id = ?;

-- 前端展示代码
this.setData({
    video: {
        ...result,
        author: result.author_name
    }
});
```

3. 评论表 (comments)

```
CREATE TABLE comments (
    id VARCHAR(32) PRIMARY KEY,
    video_id VARCHAR(32) NOT NULL,
    user_id VARCHAR(32) NOT NULL,
    content TEXT NOT NULL,
    likes_count INT DEFAULT 0,
    reply_count INT DEFAULT 0,
    parent_id VARCHAR(32),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP NULL,
    INDEX idx_video_id (video_id),
    INDEX idx_user_id (user_id),
    INDEX idx_parent_id (parent_id),
    FOREIGN KEY (video_id) REFERENCES videos(id),
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (parent_id) REFERENCES comments(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

数据插入位置及SQL示例:

1. 评论列表 (`miniprogram/pages/video-detail/video-detail.ts` 第17-39行)

```
-- 查询评论列表
SELECT
```

```
    c.*,
    u.nickname as username,
    u.avatar_url,
    (SELECT COUNT(*) FROM likes
     WHERE target_id = c.id
     AND target_type = 'comment'
     AND user_id = ?) as has_liked
FROM comments c
JOIN users u ON c.user_id = u.id
WHERE c.video_id = ?
    AND c.parent_id IS NULL
    AND c.deleted_at IS NULL
ORDER BY c.created_at DESC
LIMIT ?, ?;

-- 前端展示代码
this.setData({
    commentList: results.map(item => ({
        id: item.id,
        username: item.username,
        content: item.content,
        time: formatTime(item.created_at),
        likes: item.likes_count,
        hasLiked: item.has_liked > 0,
        replyCount: item.reply_count
    }))
});
```

[其他表的 SQL 和具体实现保持不变...]

性能优化建议

1. 索引优化

```
-- 为经常查询的字段添加复合索引
ALTER TABLE videos ADD INDEX idx_category_created (category_id, created_at);
ALTER TABLE comments ADD INDEX idx_video_created (video_id, created_at);
```

2. 分页优化

```
-- 使用 id 分页代替 OFFSET
SELECT * FROM videos
WHERE id > ? AND category_id = ?
ORDER BY id ASC LIMIT ?;
```

3. 缓存策略

```sql
-- 热门视频列表缓存
-- Redis Key: hot_videos_${category_id}
SELECT id, title, thumbnail_url
FROM videos
WHERE category_id = ?
ORDER BY views_count DESC
LIMIT 10;

-- 用户信息缓存
-- Redis Key: user_info_${user_id}
SELECT id, nickname, avatar_url
FROM users
WHERE id = ?;
```