1. **数据库服务的背景及功能需求**

数据库服务在本系统中主要使用在三个部分：1，在数据抓取子系统中将抓取的数据进行存储以实现持久化。2，在内容离线分析子系统中对数据结构化查询数据及高效的读取。3，在实时处理模型中的自我调整阶段对临时数据流进行高速存储分析。因此，在多层结构不同环境的背景下，对数据库的提出了一下三点功能需求：

1. 在离线模型中，为更加充分的进行数据分析，在数据读取时需要进行复杂的结构化查询，并获得结构化数据以便于数据分析。
2. 在数据抓取子系统中，数据的抓取量要达到十万以上的级别上切可能出现并发处理以提高数据抓取的效率，因此需要数据库能够长时间稳定的完成数据存储。
3. 在实时处理阶段，应对大量数据流数据库服务应当完成高效的存储及读取，并能实现简单的数据查询。
4. **数据库服务的概念结构设计**

本数据库服务采用两套数据库系统应对不同场景的需求。

在实时处理模型中采用Redis键值数据库来进行简单数据查询及高效的数据存储及读取。

**2.1 mysql数据库系统概念结构设计**

在数据抓取子系统及内容离线分析子系统中，根据需求采用Mysql数据库进行数据存储。在Msql数据库系统中使用的为第二范式，相较于第一范式原子性可能会差一些，通过各类为主建，成为其他表的函数依赖，其优点是在查询过程中可以避免对大量数据的查询，适用与查询次数较多的该系统。

Mysql数据库系统以表为数据的主要存储形式。

歌单模块： 

作者模块：



歌曲模块：





数据库E-R图设计



**三,数据库的逻辑结构设计**

歌单 (歌单ID，名称，作者ID，封面，创建日期，更改日期，介绍，播放次数，收藏次数，分享次数，评论次数，标签)

歌曲 (歌曲ID，歌名，流行指数，时长，介绍，歌手ID，歌曲图片，公司，发布时间，类型，评论数)

作者 (作者ID，姓名，头像，城市)

评论 (评论ID，内容，作者ID，获赞数，发布时间)

歌手 (歌手ID，姓名)

标签 (标签ID，标签名)

歌曲歌单关系表(歌单ID, 歌曲ID)

**数据字典**

歌单表（Playlist）：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 字段限制 |
| List\_Id | 歌单ID | Int | Primarykey |
| List\_Name | 歌单名称 | varchar(50) | Not null |
| Author\_Id | 作者ID | Int | Not null |
| List\_img | 歌单封面 | varchar(100) | Not null |
| List\_creatTime | 创建日期时间戳 | Bigint | Not null |
| List\_updataTime | 更改日期时间戳 | Bigint | Not null |
| Description | 介绍 | varchar(200) | / |
| PlayCount | 播放次数 | Int | Not null |
| SubscribedCount | 收藏次数 | Int | Not null |
| ShareCount | 分享次数 | Int | Not null |
| CommentCount | 评论次数 | Int | Not null |
| Tags | 标签 | Int | Not null |

歌曲表（Music）：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 字段限制 |
| Music \_Id | 歌曲ID | Int | Primarykey |
| Music \_Name | 歌曲名称 | varchar(50) | Not null |
| Singer\_Id | 歌手ID | Int | Not null |
| Music \_img | 唱片图片 | varchar(100) | Not null |
| Music \_creatTime | 歌曲发布时间 | Bigint | Not null |
| Popularity | 流行指数 | int | Not null |
| Description | 介绍 | varchar(100) | / |
| Duration | 时长(毫秒) | Int | Not null |
| Company | 公司 | varchar(50) | / |
| SubType | 演唱类型 | varchar(50) | / |
| CommentCount | 评论次数 | Int | Not null |

作者表（Author）：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 字段限制 |
| Author \_Id | 作者ID | Int | Primarykey |
| Author \_Name | 作者名称 | varchar(50) | Not null |
| city | 城市邮编 | Int | / |
| Author \_img | 作者头像 | varchar(100) | / |

评论表（Comment）：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 字段限制 |
| Comment\_Id | 评论ID | Int | Primarykey |
| Comment\_content | 评论内容 | varchar(200) | Not null |
| LikedCount | 获赞数 | Int | Not null |
| Music \_Id | 歌曲ID | Int | Not null |
| Author\_Id | 作者ID | Int | Not null |
| Comment \_creatTime | 评论发布时间 | Bigint | Not null |

歌手(Singer）：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 字段限制 |
| Singer \_Id | 歌手ID | Int | Primarykey |
| Singer \_name | 歌手姓名 | varchar(50) | Not null |

标签(Tags）：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 字段限制 |
| Tags \_Id | 标签ID | Int | Primarykey |
| Tags \_name | 标签名称 | varchar(200) | Not null |

歌曲歌单关系表(Relationsip)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 字段限制 |
| List\_Id | 歌单ID | Int | Primarykey |
| Music \_Id | 歌曲ID | Int | Primarykey |

**四、数据库的物理结构设计**

**根据以上逻辑分析所得到表的关系，使用SQL语言设计得到数据库和数据表。**

1. **创建数据库**

create database wymusic;

1. **创建表**

用户表

create table Playlist

(Id char(18) not null,

Name varchar(30) not null,

Tel char(11) not null,

Sex char(2) not null check(Sex in ('男','女')),

Age int not null check(Age>0),

Password varchar(20) not null check(len(Password)>6),

primary key(Id,Name)

)