

本科生《数据库系统原理》

系统需求规格说明书

|  |  |
| --- | --- |
| **姓 名：** | **文楷昂** |
| **学 号：** | **19281141** |
| **班 级：** | **AI1901** |
| **日 期：** | **2021年1月5日** |

目 录

[一、技术架构 2](#_Toc1173)

[二、系统性能指标 3](#_Toc19760)

[三、技术选型规划 3](#_Toc31540)

[1. 技术选型： 3](#_Toc12697)

[2. 可行性分析： 3](#_Toc1354)

[四、 系统业务分析 4](#_Toc15625)

[1. 系统业务需求 4](#_Toc30469)

[2. 系统业务场景 4](#_Toc30808)

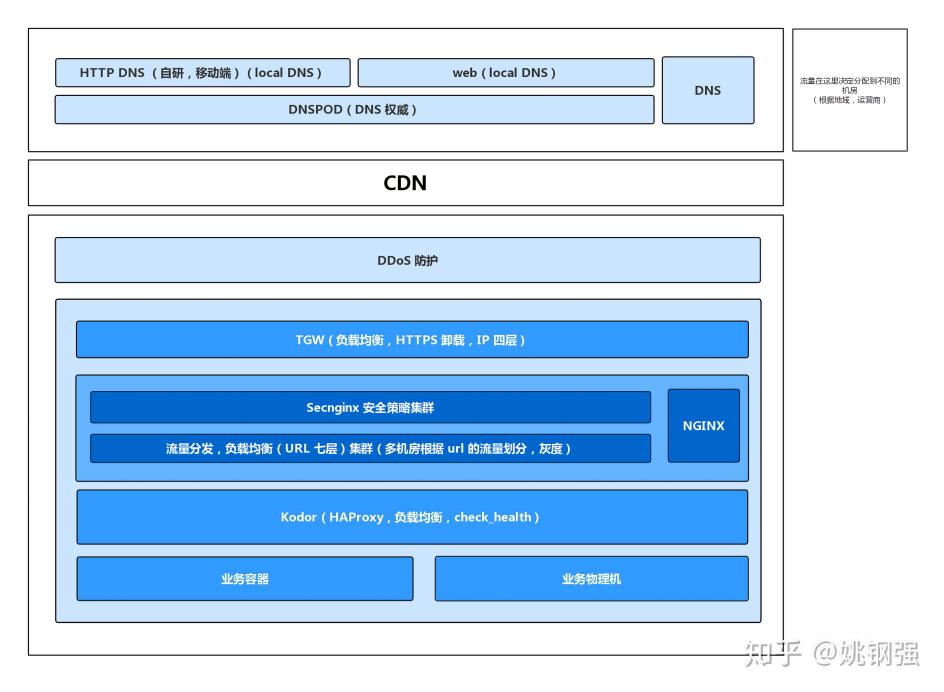
[3. 数据字典 5](#_Toc11702)

[4. 数据处理需求 9](#_Toc19461)

[数据流图： 9](#_Toc13967)

[5. 系统的非功能性需求 10](#_Toc25726)

# 一、技术架构



1. 微服务架构，知乎从 11 年就开始了微服务的探索，尝试过 protocol buffers、Avro、Thrift，最终在 16 年确认使用 Thrift，同时使用 Consul 和 HAProxy 作为注册中心和负载均衡。是在 14 年确认的这套微服务架构，并且稳定使用到了现在。所以大家不要问为什么不使用 gRPC 了。
2. 云平台，知乎有自己的内部研发的 ZAE ，绝大部分的在线业务容器在 15 年就已经全部跑在了 Docker 里，现在我们 HBase 和 Kafka 也是跑在容器里的。我们最开始使用的是Mesos 做的资源调度，现在已经切换到了 Kubernetes 。
3. 部署平台，知乎的部署平台是与 ZAE 在一起的， 基于 Jenkins 搭建的自动集成，在 MR（Gitlab） 阶段自动使用 SonarQube 进行静态代码检查。部署分为测试环境，办公室环境，金丝雀1（灰度单个容器），金丝雀2（灰度 20% 流量），生产环境（100% 流量上线）。如果金丝雀阶段出现错误，会自动进行回滚操作。
4. 监控，我们主要基于 Grafana、OpenTracing、Graphite 等搭建了监控系统。同时自研了 Halo 可以方便的是业务方观测到服务之间的依赖关系、响应时间（P95, P99, P999）、错误数。同时也进行了新技术的尝试，目前在业务容器监控使用了Prometheus 。
5. 存储，主要是 MySQL、Redis、HBase；正在调研 TiDB，目前有一套生产集群上线准备给「已读」服务使用。
6. 消息队列，早期使用自研的 Sink，目前使用 Kafka，同时提供在 Kafka 的基础上包装了Beanstalkd 作为任务队列方便业务进行使用。
7. 编程语言，Python、Golang、Java、Rust。目前 Golang 使用比较广泛，Python 使用场景逐渐变少。Java 在一些算法项目和商业系统中有使用。搜索系统使用的 Rust 重写 Lucene，现并在此基础上重写了类 ES 的集群化搜索引擎。架构师 @孙晓光 会去 https://rustcon.asia/ 做分享。
8. 目前已经搭建好完备的测试环境，会在 Q2 投入使用。
9. 现在已经搭建了多机房，目前重要的业务都是异地多活。

# 二、系统性能指标

1. 响应时间： 响应时间在秒级别，正常情况下1秒内能给出响应。
2. 吞吐量：1000个请求/秒
3. 资源使用率：

内存：和数据规模正相关,维持基本运行应在100MB左右。

磁盘I/O：每秒1000个请求，平均一个请求50kB，需要50Mb /秒的磁盘IO

网络I/O：同磁盘，50Mbps

cpu占用：以上述数据规模，估计需要8核\*2Ghz的CPU可以满足

1. 并发用户数：500

# 三、技术选型规划

技术选型：

对于千万级别数据，可以不进行分表。具体数据估算过程如下：

根据网上查询资料，知乎每日新增问题约为五千个，每个问题的平均回答数约为10个。按照数据库存储一年365天的问题估算，问答数据表的记录数估计为：5000\*365\*10=18250000条，即千万级别。

对于用户表，已知知乎新增用户数量约为57万/月，则用户表记录条数每年新增约为570000\*24=13680000条，也是千万数量级。

对于千万级别并发，需要使用多台服务器并进行分库。

由于直接面向用户，需要进行前后端分离的设计，在数据库系统中即采用MVC三层架构。

数据库端使用Ubuntu系统下的mysql数据库，实现前后端分离。

Web端使用基于Python语言的Django框架，以mysql为数据库，bootstrap为前端框架，以网页作为主要交互方式。现有技术和框架可以支持预想的功能实现。

可行性分析：

问题在于mysql对于百万级别的数据量和并发处理速度有限，需要对相关语句进行细致的优化。但是从网络信息可知，在使用复合索引时不分表也可实现千万级数据量的查询。所以总的来说，该系统具有技术可行性。

# 系统业务分析

## 系统业务需求

早期的知乎是一个聚焦专业话题的小众社区，主要面向互联网圈内人士。实行社区开放政策后，知乎转变为一个面向广大网友的知识分享平台。以问答社区为核心基础，问答内容不仅有各类专业知识，还包括了大众话题、娱乐话题。同时也推出知乎Live、私家课、值乎等多种形态的知识付费业务，用户主要是发达城市有学习需求和自我提升意识的人群。

所做的解惑论坛，同样以论坛社区为核心基础，允许用户通过网页随时随地访问。

用户首先需要注册、登录后，才被允许访问论坛。登录后，用户可以：

在首页浏览公告、今日推荐；

搜索自己感兴趣的内容；

根据发布时间、回复数量、分类来筛选帖子；

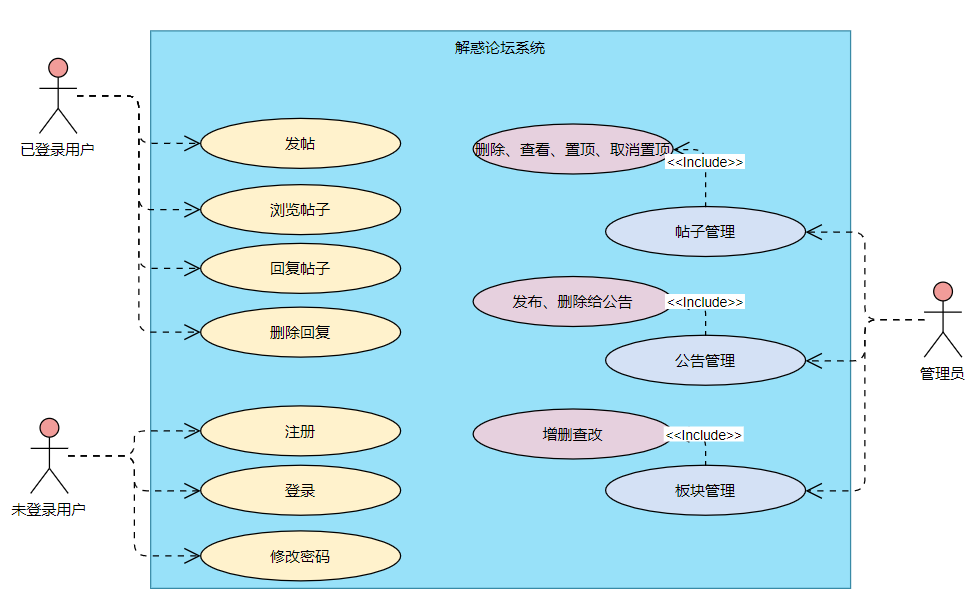
发布帖子、修改自己发布的帖子；

查看评论区、发表评论、删除自己的评论；

退出解惑论坛。

等。帖子可以按照其分类（全部、数据库、深度学习等）、回复数量、发布时间进行筛选。点开一个帖子后，展示该帖子的图文内容，发布时间，类别，作者及所有评论。

除面向用户的业务部分，还配有管理端，供管理员进行帖子、公告、板块分类三个模块的管理。在帖子管理模块，可以对帖子进行删除、置顶推荐、取消置顶等；在公告管理模块，可以发布、删除公告；在板块管理模块，可以添加、查看、删除分类板块。



## 系统业务场景

本系统一共有注册、登录、修改密码、公告栏、论坛广场、发布帖子、浏览帖子、评论、帖子管理、公告管理、板块管理11个业务场景。

对于跨用户的交互场景，我设定了一定限制：只有管理员可以修改和删除其他人的帖子、回复，一般用户只能修改自己的密码、帖子和回复等。

## 数据字典

### 数据表结构

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **表名** | **App01\_announcement** |  |  |  |  |  |
| **数据库用户** | **root** |  |  |  |  |  |
| **主键** | **id** |  |  |  |  |  |
| **外键** |  |  |  |  |  |  |
| **排序字段** | **id** |  |  |  |  |  |
| **索引字段** | **id** |  |  |  |  |  |
| **字段名称** | **数据类型** | **说明** | **允许为空** | **唯一** | **默认值** | **约束条件** |
| **id** | **int** |  | **N** | **Y** |  | **主键，自增** |
| **A\_title** | **varchar(64)** | **公告标题** | **N** | **N** |  |  |
| **A\_content** | **varchar(3000)** | **公告内容** | **Y** | **N** |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **表名** | **App01\_kind** |  |  |  |  |  |
| **说明** | **板块分类** |  |  |  |  |  |
| **数据库用户** | **root** |  |  |  |  |  |
| **主键** | **id** |  |  |  |  |  |
| **外键** |  |  |  |  |  |  |
| **排序字段** | **id** |  |  |  |  |  |
| **索引字段** | **id** |  |  |  |  |  |
| **字段名称** | **数据类型** | **说明** | **允许为空** | **唯一** | **默认值** | **约束条件** |
| **id** | **int** |  | **N** | **Y** |  | **主键，自增** |
| **K\_name** | **varchar(16)** | **类别名** | **N** | **N** |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **表名** | **App01\_reply** |  |  |  |  |  |
| **说明** | **回复** |  |  |  |  |  |
| **数据库用户** | **root** |  |  |  |  |  |
| **主键** | **id** |  |  |  |  |  |
| **外键** | **R\_tid、R\_uid** |  |  |  |  |  |
| **排序字段** | **id** |  |  |  |  |  |
| **索引字段** | **Id、r\_tid** |  |  |  |  |  |
| **字段名称** | **数据类型** | **说明** | **允许为空** | **唯一** | **默认值** | **约束条件** |
| **id** | **int** |  | **N** | **Y** |  | **主键，自增** |
| **R\_tid** | **varchar(16)** | **帖子id** | **N** | **N** |  |  |
| **R\_uid** | **varchar(16)** | **回复人id** | **N** | **N** |  |  |
| **R\_photo** | **varchar(128)** | **图片路径** | **Y** | **N** |  |  |
| **R\_time** | **Date** | **回复时间** | **N** | **N** |  |  |
| **R\_content** | **Varchar(256)** | **回复内容** | **N** | **N** |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **表名** | **App01\_topic** |  |  |  |  |  |
| **说明** | **帖子** |  |  |  |  |  |
| **数据库用户** | **root** |  |  |  |  |  |
| **主键** | **id** |  |  |  |  |  |
| **外键** | **T\_uid,t\_kind** |  |  |  |  |  |
| **排序字段** | **id** |  |  |  |  |  |
| **索引字段** | **id** |  |  |  |  |  |
| **字段名称** | **数据类型** | **说明** | **允许为空** | **唯一** | **默认值** | **约束条件** |
| **id** | **int** |  | **N** | **Y** |  | **主键，自增** |
| **t\_uid** | **varchar(16)** | **发帖人** | **N** | **N** |  |  |
| **T\_kind** | **varchar(32)** | **类型** | **N** | **N** |  |  |
| **Create\_time** | **Date** | **发帖时间** | **N** | **N** |  |  |
| **T\_photo** | **varchar(128)** | **图片路径** | **Y** | **N** |  |  |
| **T\_content** | **Varchar(3000)** | **帖子内容** | **N** | **N** |  |  |
| **T\_title** | **Varchar(64)** | **帖子标题** | **N** | **N** |  |  |
| **T\_introduce** | **Varchar(256)** | **帖子介绍** | **N** | **N** |  |  |
| **recommend** | **tinyint** | **是否推荐** | **N** | **N** | **0** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **表名** | **App01\_user** |  |  |  |  |  |
| **说明** | **用户** |  |  |  |  |  |
| **数据库用户** | **root** |  |  |  |  |  |
| **主键** | **id** |  |  |  |  |  |
| **外键** |  |  |  |  |  |  |
| **排序字段** | **id** |  |  |  |  |  |
| **索引字段** | **id** |  |  |  |  |  |
| **字段名称** | **数据类型** | **说明** | **允许为空** | **唯一** | **默认值** | **约束条件** |
| **id** | **int** |  | **N** | **Y** |  | **主键** |
| **uid** | **varchar(16)** | **用户名** | **N** | **N** |  |  |
| **password** | **varchar(32)** | **密码** | **N** | **N** |  |  |
| **Create\_time** | **Date** | **注册时间** | **N** | **N** |  |  |

### 数据结构

数据结构：公告

含义说明：由管理员管理的公告，会出现在论坛首页。

组成：id，标题，公告内容

数据结构：帖子类型对应表

含义说明：是“解惑”论坛系统的次体数据结构，定义了一个类型编号和类型的关系。

组成：类型id，类型文本

数据结构：回复

含义说明：是“解惑”问答论坛系统的主体数据结构，定义了评论区的有关信息。

组成：回复id，所回复的帖子的id，回复人id，图片路径，回复时间，回复内容。

数据结构：帖子

含义说明：是“解惑”问答论坛系统的主体数据结构，定义了帖子的有关信息。

组成：帖子id，帖子介绍（相当于副标题），帖子内容，发帖人，类型，图片路径，是否推荐（被推荐的帖子会显示在网站首页，公告栏下方）。

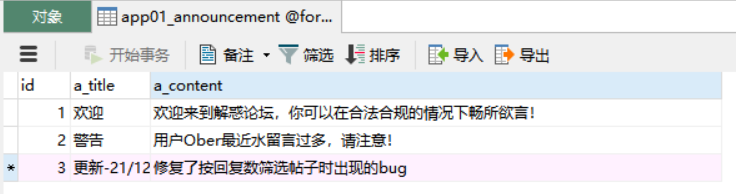
数据结构：用户

含义说明：是“解惑”问答论坛系统的主体数据结构，定义了用户的有关信息

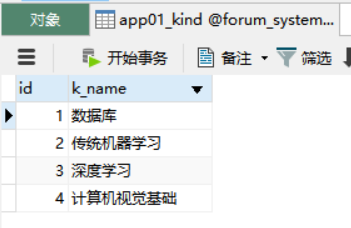
组成：用户id，用户昵称，用户密码，注册时间

### 3. 数据存储

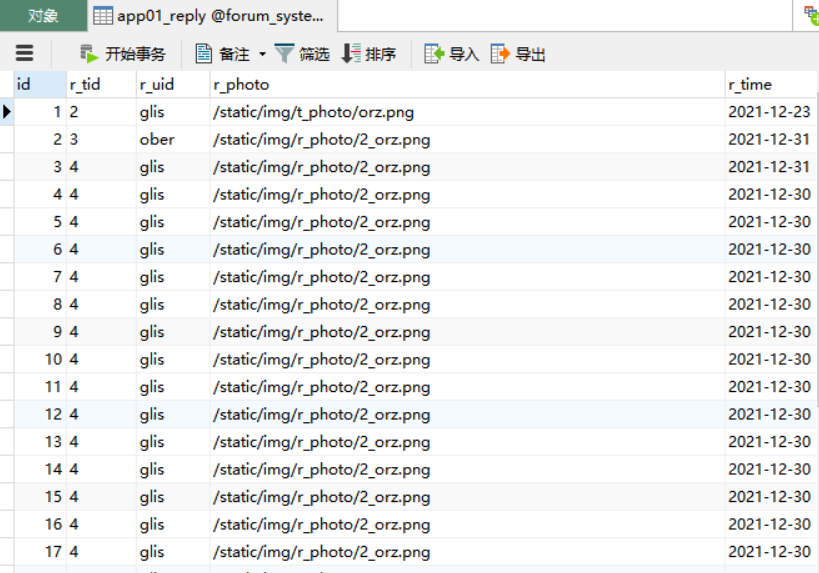
#### 3.3.1 公告表



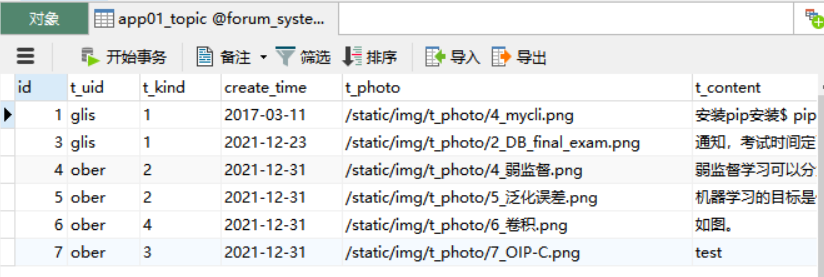
#### 3.3.2 类型表



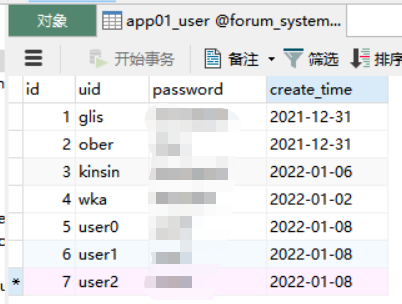
#### 3.3.3 回复表



#### 3.3.4 帖子表



#### 3.3.5 用户表



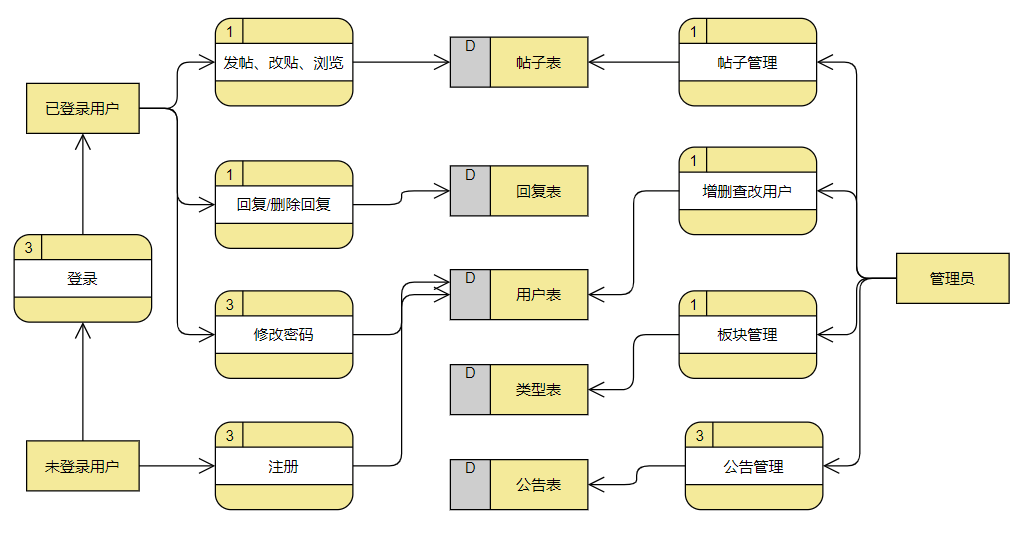
### 4.数据的存储（持久化）需求

数据持久化就是将内存中的数据模型转换为存储模型,以及将存储模型转换为内存中的数据模型的统称. 数据模型可以是任何数据结构或对象模型,存储模型可以是关系模型、XML、二进制流等。cmp和Hibernate只是对象模型到关系模型之间转换的不同实现。

“解惑”问答论坛系统中，帖子表、类型对应表和回复表有永久的存储需求，用户表则可以以两年为存储期限，删除两年未曾登录的用户信息。

## 数据处理需求

数据流图：



## 系统的非功能性需求

### 5.1 业务处理性能需求

1. 由系统规划与可行性分析报告指出，查询页面响应时间需要保持在5秒以内，同时需要具备处理百万级别并发的能力。

2. 安全性需求：分为程序安全、系统安全、数据安全。程序安全是指开发的程序是安全的，程序上没有安全的漏洞，例如Web开发中服务器代码没有对输入的参数进行验证，从而导致客户端机器人轻易的获取数据。系统安全指的是系统整体的安全，例如安全的粒度，未经授权的用户是否可以轻易的访问非法的数据等。数据的安全是对数据的保护，数据库中数据有没有做审核，用户之间是否会共享数据等。

3. 完整性需求：指为完成业务需求和系统正常运行本身要求而必须具有的功能。包括数据备份、恢复、日志管理、垃圾数据清除等基本功能、用户管理等。

### 5.2 安全性

程序安全：是指开发的程序是安全的，程序上没有安全的漏洞，例如Web开发中服务器代码没有对输入的参数进行验证，从而导致客户端机器人轻易的获取数据。

系统安全指的是系统整体的安全，例如安全的粒度，未经授权的用户是否可以轻易的访问非法的数据等。

数据的安全是对数据的保护，数据库中数据有没有做审核，用户之间是否会共享数据等。

在“解惑”论坛系统中，用户需要以自己的账号和密码登录后才能进行操作，保障了系统的安全性；

一名用户无法修改、删除其他用户的帖子、回复、密码等，保障了数据的安全性。

### 5.3 完整性

1. 用户、帖子、回复、公告、类型五个数据表的主键均为对应id；
2. 帖子有2个外键：用户id和类型id；回复有2个外键：用户id、帖子id；
3. 帖子、回复、用户注册的时间需要为合法时间，为DATETIME类型，以固定的格式：YYYY-MM-DD-hh:mm:ss 存储；
4. 帖子、回复的内容必须符合互联网审查制度，不能出现非法/有害内容。管理员将定期对内容进行审核，不过审的会被ban，封禁用户或删除帖子，且在公告栏公示；
5. 帖子、回复的内容需要符合字数限制，不能过长：
6. 不得灌水。一旦被管理员发现灌水，将可能给予警告、封号处罚。处罚同样会在公告栏公式；
7. 帖子、回复的内容可以附带图片，同时其也需符合第3条的合规性限制；
8. 分类表需预置足够量的分类信息，每个帖子可以分到多个分类，且分类表需要能够动态更新、升级。