数据库的设计

之前所做的前置程序没有采用数据库，而是采用改变客户端向服务端所发送消息的前缀与后缀，并且在服务器端维护一个user\_name，socket的哈希表来确定所需要发送到的用户。

那么，接下来进行数据库设计：

两个实体: 用户和消息

1. 用户（User）

程序中采用的对每个用户的识别依据仅为 “用户名”，很明显这在正常的聊天通信程序中并不合理，因为 用户名有可能相同，那么我们就需要 唯一标识码 来识别用户，并且要包括用户的完整信息，并且将用户群组看作一个特别的用户

* 1. 编号—标识码表
     1. 编号
        1. 起始于 0
        2. 地增量 1
     2. 标识码（账号）
        1. 随机生成 12 位数
           1. 需要事先生成
           2. 使用时伪随机获取
        2. 拟实现类似git回退版本只用输入前几位就可以作为账号然后和密码一起被查询
           1. 需要设定查询数的上限值，一旦超过某个值如20，则会要求再输入一位甚至更多
           2. 当查询返回个数小于某个值如 10 的时候， 则会去检查所登录ip地址与常用ip地址是相同，（因为只输前几位账号是为了万一遗忘，而不是为了给危险分子有机可趁）
           3. 所以要实现一个 标识码-ipv4地址表-ipv6地址表
           4. 可能会面对的困难

个人ip地址可以随意改变，如何确定准确的ip地址

* + 1. 创建时间
  1. 用户信息表
     1. 用户编号
     2. 对应 编号—标识码表 的编号
     3. 密码
     4. 姓名
     5. 昵称
     6. 地址表编号
        1. 地址表
           1. 国家

国家表

国家编号

国家名称

* + - * 1. 省/直辖市/州/其他

省/直辖市/州/其他 表

编号

名称

* + - * 1. 城市

城市表

编号

名称

* + - * 1. 详细地址
    1. 喜欢的聊天方式
       1. 聊天方式表
          1. 聊天方式编号
          2. 聊天方式内容
    2. 头像
       1. 头像表
          1. 头像编号
          2. 头像图片
    3. 状态
       1. 用户状态表
          1. 用户状态编号
          2. 用户状态内容
          3. 用户状态修改时间
    4. 创建时间

1. 好友表
   1. 编号
   2. 自己ID
   3. 好友ID
   4. 备注昵称
   5. 添加方式
      1. 添加方式表
         1. 添加方式编号
         2. 添加方式内容
   6. 好友被分组别
      1. 分组表
         1. 编号
         2. 自己ID
         3. 组别名称
         4. 容量
         5. 创建时间
2. 消息表
   1. 编号
   2. 类型
      1. 消息类型表
         1. 编号
         2. 消息类型名称
   3. 内容
   4. 来自ID
   5. 发向ID
   6. 接收状态
   7. 发送时间
3. 群组（金星， 木星， 水星 etc.）
   1. 编号
   2. 分类
      1. 群组类别表
         1. 编号
         2. 名称
   3. 群组讨论话题
      1. 群组话题表
         1. 编号
         2. 群组话题内容
         3. 热度
   4. 群用户
      1. 群用户表
         1. 编号
         2. 群ID
         3. 用户ID
         4. 群内名称
         5. 创建时间