# 杭州电子科技大学

## 硕 士 学 位 论 文

**题　　目：** 电动自行车智能充电系统设计实现及关键技术研究


**研 究 生** 周雪梅

**专　　业** 电子科学与技术

**指导教师** 陈科明 副教授



**完成日期** 2020 年 3 月

杭州电子科技大学硕士学位论文

# 电动自行车智能充电系统设计实现及关键技术研究

研 究 生：　　周雪梅

指导教师：　陈科明 副教授

2020 年 3 月

**Dissertation Submitted to Hangzhou Dianzi University**

**for the Degree of Master**

# Design and Key Technology Research of Intelligent Charging System for Electric Bicycle

Candidate: Zhou Xuemei

Supervisor: Chen Keming, Associate Professor

**Mar，2020**

# 杭州电子科技大学

# 学位论文原创性声明和使用授权说明

## 原创性声明

本人郑重声明： 所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

论文作者签名： 日期： 年 月 日

## 学位论文使用授权说明

本人完全了解杭州电子科技大学关于保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属杭州电子科技大学。本人保证毕业离校后，发表论文或使用论文工作成果时署名单位仍然为杭州电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。（保密论文在解密后遵守此规定）

论文作者签名： 日期： 年 月 日

指导教师签名： 日期： 年 月 日

# 摘要

在互联网+的大背景下，电动自行车充电桩朝着智慧、互联的方向转型。市场上出现了很多联网式充电桩，通过设备联网，提供多样化的智慧服务。但是现有联网式充电系统存在一些问题：在提供邻近充电桩查询服务时，大多根据用户与充电桩之间的距离进行筛选，在数据量大的情况下效率低；充电系统对于高并发访问支持不足，高峰时段服务不稳定。

本文设计实现了一套电动自行车智能充电系统，并对以上两个问题进行深入研究，提出了解决方案。系统包括硬件主控板设计、GPRS 通信模块设计、云平台搭建及应用软件开发。硬件部分设计有过流、过载、短路保护功能，实现安全充电；根据不同的人员角色，开发了移动用户端、移动运维端和 Web 管理平台，分别满足充电用户、运维人员及企业管理员的需求。

对邻近充电桩搜索方法进行优化，引入空间查询，在现有空间索引的基础上进行改进，设计了基于四叉树网格划分的邻近充电桩快速搜索算法，并改进 GeoHash 编码得到 B-GeoHash 编码，对充电桩进行编码，实现坐标数据降维。搜索时分为剪枝和过滤两个阶段，剪枝阶段根据目标近似形体与网格的空间关系，利用 B-GeoHash 编码前缀一致的特性，从大量数据中快速筛选出可能的目标集合；过滤阶段用球面距离对目标集合进行精准过滤。经过测试，该算法在 30 万数据量时也可以 1 秒内得到结果，所用时间比计算两点间距离的常规筛选方法缩短了 84%。

针对高并发情况下大量重复的数据库操作造成系统响应速度降低的问题，从服务器缓存入手，设计了一种改进的 JDBC 查询缓存机制。对 JDBC 接口进行扩展，将 SQL 结果集缓存在业务服务器的内存中，下次请求时，对请求进行过滤，如果缓存中存在对应结果集则直接读取，避免重复的数据库操作，减轻了系统压力。对缓存容量达到阈值时的缓存置换策略进行研究，改进了 LRU 算法，提高了缓存命中率。同时设计了缓存一致性维护策略，保证缓存数据的准确。改进的 JDBC 缓存机制减少了服务器连接数据库的次数，有着更高的缓存命中率，实验表明，该机制可以将系统的响应速度提高一倍，具有良好的性能。

**关键词：**智能充电系统，空间查询，B-GeoHash 编码，JDBC 缓存，缓存置换

# ABSTRACT

With the development of the Internet, electric bicycle charging pile industry is transforming towards the direction of wisdom and interconnection. There are many networked charging piles on the market, providing a variety of services through equipment networking. However, there are some problems in the existing networked charging system: when providing the proximity charging pile inquiry service, the method most of these systems token are based on the distance between the user and the charging pile, the efficiency is low in the case of a large amount of data. In the meantime, these charging systems have poor performance in the case of high concurrent access, and cannot ensure stable service during peak hours.

This thesis designs and implements an intelligent charging system for electric bicycles, and conducts in-depth research on the above two problems and proposes solutions. The system includes hardware design, GPRS communication module design, cloud platform construction and application software development. The hardware part is designed with overcurrent, overload and short circuit protection functions to achieve safe charging. According to different roles, the system develops the user version of WeChat webpage system, operator version WeChat webpage system and web management system to meet the needs of charging users, operator and business administrators.

The neighboring charging pile search method in user service is optimized. The thesis introduces spatial query and makes some improvements to existing spatial index methods, designs the fast search algorithm based on quadtree partitioning. Learning from the GeoHash coding, the thesis develops B-GeoHash coding to encode charging piles, reduces the dimensionality of the data. The algorithm is divided into two stages: pruning and filtering. In the pruning stage, according to the spatial relationship between the MBR and the target, uses the characteristics of the B-GeoHash coding to filter out possible target sets from a large amount of data quickly, then uses the spherical surface distance to filter. After testing, the algorithm can achieve results in 1 second when dealing with 300,000 data, which is 84% shorter than the conventional method.

Aiming at the problem of slow response caused by a large number of repeated database operations in high concurrency condition, a kind of JDBC-based query caching mechanism is designed. Through extending JDBC interface, the SQL ResultSet is cached in the memory of the service server. When the next request is sent, filters it before connecting to database. If the corresponding ResultSet exists in the cache, reads it directly, avoiding frequent repeated database operations and reducing system pressure. Research on the cache permutation strategy when the cache capacity reaches the threshold, improves the LRU algorithm, and increases the cache hit rate.

At the same time, the cache consistency maintenance strategy is designed to ensure the accuracy of the cache. The JDBC caching mechanism reduces the times of connecting to database. Experiments show that this mechanism can double the response speed of the system.

**Keywords**: Intelligent charging system, spatial query, B-GeoHash coding, JDBC cache, cache permutation

# 目录

摘要.....I
ABSTRACT.....II
第一章 绪论.....1
1.1 研究背景及意义.....1
1.2 国内外研究现状.....2
1.2.1 电动自行车充电系统研究.....2
1.2.2 空间查询技术研究.....3
1.2.3 缓存技术研究.....4
1.3 研究内容及章节安排.....5
第二章 电动自行车智能充电系统设计及实现.....7
2.1 系统设计.....7
2.1.1 需求分析.....7
2.1.2 整体框架.....7
2.2 硬件设计.....9
2.2.1 MCU 设计.....9
2.2.2 GPRS 通信模块设计.....11
2.2.3 通信协议设计.....13
2.3 云平台设计.....14
2.3.1 设备云.....15
2.3.2 业务云.....15
2.4 客户端设计.....18
2.4.1 客户端技术架构.....19
2.4.2 移动用户端.....21
2.4.3 移动运维端.....24
2.4.4 Web 管理平台.....26
2.4.5 页面响应速度优化.....29
2.5 系统展示.....30
2.5.1 硬件展示.....30
2.5.2 数据采集.....31
2.5.3 多种充电方式.....31
2.5.4 邻近充电桩搜索及导航.....33

2.5.5 应用展示.....34
2.6 本章小结.....34
第三章 基于四叉树网格划分的邻近充电桩快速搜索算法.....36
3.1 B-GeoHash 编码设计.....37
3.2 基于四叉树网格划分的快速搜索方法.....42
3.2.1 相关参数和概念.....43
3.2.2 剪枝阶段.....44
3.2.3 过滤阶段.....45
3.3 算法分析.....46
3.4 测试分析.....47
3.4.1 准确性测试.....47
3.4.2 耗时测试.....48
3.5 本章小结.....49
第四章 基于 JDBC 的数据缓存研究及实现.....51
4.1 JDBC 原理及实现机制.....51
4.2 JDBC 缓存设计及实现.....53
4.2.1 缓存对象.....54
4.2.2 缓存存储与查找.....54
4.2.3 缓存置换.....57
4.2.4 一致性维护.....61
4.3 测试分析.....62
4.3.1 命中率测试.....62
4.3.2 响应时间测试.....63
4.4 本章小结.....64
第五章 总结与展望.....66
5.1 主要工作和结论.....66
5.2 研究展望.....66
致谢.....68
参考文献.....69
附录.....72

# 第一章 绪论

## 1.1 研究背景及意义

电动自行车从面世到现在发展了二十多年，已经成为我国重要的交通工具，在市场上占有很大份额。如图 1.1 所示，根据工业和信息化部、中国汽车工业协会统计的数据，2018 年全国交通工具产量中电动自行车排名第二[1]。

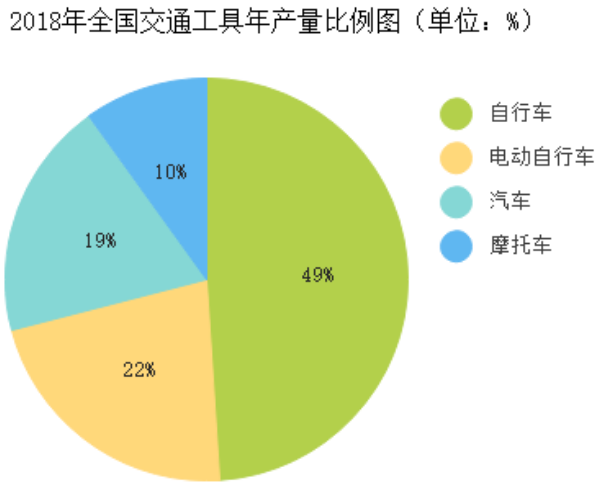



图1.1　2018 年全国交通工具年产量比例图

2009 年至 2018 年我国电动自行车社会保有量稳步上升[2]，统计数据如图 1.2，2016 年超过 2.5 亿辆，2018 年接近 3 亿辆。

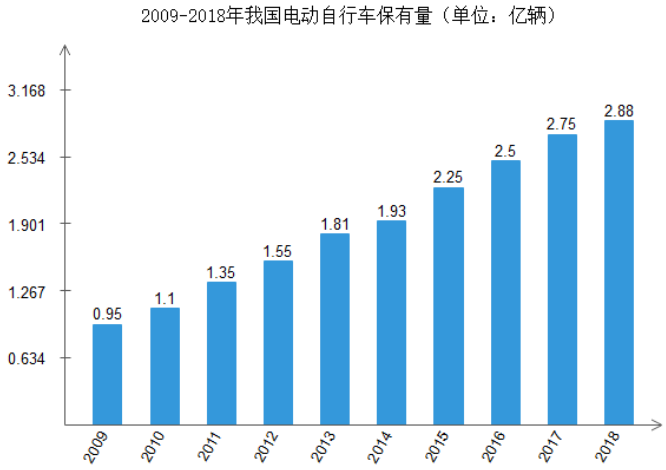



图1.2　2009-2018 我国电动自行车保有量

接近 3 亿的车辆保有量带来每天至少 1 亿次的充电需求。以往，很多居民会私拉电线进

行充电，或者拆卸电池放在房屋中充电，这些行为都带来极大的安全隐患。近几年因电动自行车电气线路老化、充电线路过负荷等原因造成的火灾事故频发，带来人员伤亡和财产损失，这引起了政府的高度重视。2017 年 12 月公安部就已发布“规范电动车停放充电加强火灾防范”的通告[3]，强调要加强对电动车充电的规范管理。杭州政府也出台政策，规定“居住出租房屋必须建立符合消防安全技术防范要求的电动自行车集中停放点或设置电动自行车智能充电桩”[4]。

充电服务是支撑电动自行车出行的必要条件，充电设施的发展影响电动自行车行业的发展质量和规模。电动自行车充电桩企业通过在区域内建立集中充电站，为居民提供规范化充电服务。早期，市面上的充电桩以投币式和刷卡式为主，充电桩只是一个充电设备，功能单一。近年来，物联网快速发展，各个企业纷纷推出联网式电动自行车充电桩，将设备入网的同时为用户提供多种交互服务，受到了人们的广泛欢迎，也符合时代发展的趋势。

但是，目前市场上的联网式电动自行车充电桩智能化程度不够，存在充电过程不安全、用户服务不完善、系统性能差等问题，无法对人们形成很强的吸引力。因此，设计一种能够保障充电安全，且具有远程控制、智能管理、报警监控等功能的电动自行车充电服务系统十分有必要。实现系统功能的同时，要保证系统的高可靠高可用，这样才能增加用户粘性，真正起到解决充电乱象的作用，实现社会效益。

本文设计实现了一套电动自行车智能充电系统，同时对两个方面的优化进行重点研究，以提升系统性能：（1）用户服务中的邻近充电桩搜索，充电桩数量越多，从中筛选出所需数据的时间越长，为了不影响用户体验，需要设计高性能的搜索算法替代常规的计算两点间距离的过滤方式；（2）频繁的数据库操作，大量用户同时访问系统，并且伴随相同数据库操作时，会带来很多开销，影响系统性能，需要设计相关机制减小重复的资源消耗。

## 1.2 国内外研究现状

### 1.2.1 电动自行车充电系统研究

随着政府对绿色出行的大力推广，电动自行车行业得到很大发展。作为配套的支撑设施，电动自行车充电桩行业也发展迅速。在互联网+的大背景下，电动自行车充电桩服务商朝着智能、互联的方向转型。目前国内市场上，驴充充智能充电桩对接了云平台，但是主机仍然是投币刷卡式的，缺少配套的用户软件服务；九牛科技的云智充开发手机 APP，为用户提供自助充电、在线支付等服务，但是需要下载 APP，给用户造成了使用负担；北京小绿人科技推出的智能电动自行车充电桩产品，用户需要自己选择充电电压，不能涵盖所有的用户电池类别，并且不少用户反映电池被充坏的情况。从以上市场调研内容可以看出，虽然电动自行车充电桩发展已进入互联网时代，但因为很多厂商急于将产品推向市场，导致设计不周全、功

能不完善，很多充电桩仅实现了基本的互联功能，在系统性能、用户服务上还有很大改进空间，并不能达到一个成熟产品的标准。

国外的电动交通工具中，电动汽车一直是主流，并且经过很多年的发展，已经形成了较为规范的行业体系[5]。电动自行车充电桩大多是和电动汽车充电桩合并在一起，据统计，目前美国投入使用的充电桩已经超过 5 万个[6]。但是电动汽车和电动自行车的电池存在很大差异，将两种充电桩统一设计难度大，并且我国电动汽车和电动自行车的发展不均衡，电动自行车的数量远远大于电动汽车的数量，设计一体化的充电桩不符合我国国情。

结合研究和调研，总结出当前电动自行车充电桩行业存在的一些问题：

（1）硬件落后，目前很多充电桩的硬件设计存在不足，无法保证充电过程的安全；不合理的充电方式会对电池造成损伤，减少电池使用寿命。

（2）配套软件设计落后，没有充分考虑不同职责的人的需求。除了有充电需求的用户，运维人员、企业管理员等同样需要应用软件服务，来更好地管理和运维系统。

（3）业务平台设计不完善，在数据量大和高并发情况下性能较差。随着企业业务的扩张，充电桩数量会迅速增加，当为用户提供邻近充电桩查询的时候，大量的数据会造成处理时间过长，用户迟迟得不到响应，体验感差。同时，电动自行车充电桩使用存在高峰期，如公司附近的充电桩早晨上班时段是使用高峰期，小区附近的充电桩下班、周末时段是使用高峰期。在这些时段，大量用户同时接入系统，会给系统带来很大压力，造成服务不稳定甚至出现故障。

### 1.2.2 空间查询技术研究

充电桩的经纬度位置是地理信息系统 GIS（Geographic Information System）[7]的一部分，为用户提供邻近充电桩搜索服务时，借助 GIS 的空间查询技术，可以大大提升搜索效率。

空间查询是对空间数据的查询，支持点、线、多边形等几何数据类型。空间索引是实现空间查询的方式，空间索引将空间对象的位置、形状等空间关系按一定的顺序排列，提高数据检索效率[8]。

空间索引技术在国外发展比较成熟。J.L.Bentley 和 R.A.Finkel 于 1974 年一起提出了存储空间多维点的四叉树索引，通过对区域不断地进行四分构建索引。Jon Bentley 在 1975 年提出 KD-tree[9]，对多维空间进行分割，实现对多维空间数据的搜索，是一种特殊的空间二分树。John T. Robinson 在 1981 提出的 KDB-tree 在 KD-tree 的基础上结合了 B 树的特性，以 B 树的方式进行插入和删除，对点有较好的检索效率，对空间线、面等区域范围要素检索效率略显不足。1984 年 Antonin Guttman 提出 R 树索引，设计用空间对象的最小边界矩形 MBR 来近似表示该对象。Benkman 对 R 树的插入处理进行优化，设计了 R*树。1996 年 Seeger 等又提出区域重叠和转换结合的改进搜索方法[10]。

国内的空间索引技术也在不断发展。史杏荣、孙贞寿等提出一种基于固定网格划分的四

分树空间索引机制-CELLQTREE[11]，有效减少了空间查询的检索范围。陈敏、王晶海等提出了一种改进的 R*-树空间索引结构，改进后的 R*-树与原始的 R*-树相比具有更高的性能[12]。张军旗、周向东等提出了一种基于聚类分解的高维度量空间 B~+-Tree 索引，通过聚类分解，对数据进行更细致的划分来减少查询的数据访问[13]。申丹丹提出一种 HBase 空间索引方案，利用 Geohash 编码,构建空间索引[14]。

总之，目前发展出的空间索引技术很多，各有优缺点，可根据具体需求和实际应用场景，选择合适的方法，再加以改进。

### 1.2.3 缓存技术研究

互联网中有一个关于网络价值的重要定律——梅特卡夫定律，阐明网络的价值与该网络内的用户数量成正比[15]。用户数量与用户体验有关，用户体验受页面响应时间影响。页面响应时间有一个 2/5/10 秒的标准划分[16]，当页面加载时间小于 2s，用户会认为系统有吸引力；响应时间在 2-5s 内，用户会觉得还不错；但当一个页面 10s 还没有响应，那么用户的体验是糟糕的，大部分用户会放弃使用该网站。用户的流失将会导致网站价值下降，对于我们的智能充电服务系统来说，就是企业效益和社会效益的降低。我们在实现充电功能的同时，也要注重提升系统功能，通过良好的使用体验吸引更多的用户。

在页面响应速度优化方面，现有的一些通用方式如资源压缩、CDN 加速等，可以带来性能提升，但仅仅有这些是不够的，如果想要在市场上取得优势，必须寻求更多的优化，使用缓存是一个重要的优化方向。

缓存通过减少应用程序对系统物理数据源访问量和频次提高计算机性能[17]。请求结果存储在内存中，当数据没有发生改变时，下次请求直接从缓存区读取数据，无需重新建立连接，减小了资源消耗。当前的软件应用系统缓存大致可分为三类：浏览器缓存、服务器缓存和数据库缓存。

大部分前端应用都是基于浏览器的，浏览器缓存遵循 http 协议的相关设置将页面资源如图片、cookie 文件等缓存在磁盘中[18]。在资源有效期内，重新打开页面，从本地读取资源，无需请求服务器，避免了建立网络连接的开销，加快了页面响应速度。但是浏览器缓存容易受到用户的影响，如果用户禁止浏览器缓存，就无法通过此方式进行性能优化。且浏览器缓存对不同浏览器是不一样的，用户更换浏览器，就失去了作用。另外我们要慎重使用浏览器缓存，避免带来网站无法及时更新的问题。

服务器程序是业务的重要支撑，是数据交互、逻辑处理的核心。服务器缓存将满足要求的数据存储在业务服务器中，通过减小请求路径长度提升系统业务处理效率[19]。用户进行交互操作时，客户端向服务器发起请求，服务器处理得到数据后回复给客户端，同时把数据缓存到自有内存中。下次用户进行相同的操作，就可以直接从缓存中取出数据回复给用户。目前比较流行的服务器缓存技术有 Memcached 和 Redis。

Memcached 是由 Danga Interactive 公司开发的一款高性能的分布式内存系统[20]，最初是为了加快 LiveJournal 项目的访问速度。被很多大型网站用来应对高负载情况，如 Wikipedia、Digg、Slashdot[21]。Memcached 的基本工作流程为：先检查 Memcached 中是否存在请求数据，若存在，则直接返回，不对数据库进行操作；若不存在，则连接数据库，处理请求得到结果返回给客户端，同时缓存一份数据到 Memcache 中。数据库更新时同步更新 Memcached 中的数据，保证数据一致性。

Redis（Remote Dictionary Server）是一个采用 key-value 数据模型的 NoSQL 数据库[22]，支持 String、List、Hash、Set、Sorted Set 等多种数据类型，通过把数据保存在内存中或者使用虚拟内存技术来提高系统的使用效率[23]。Github、Twitter、新浪微博等公司都在使用 Redis[24]。

Memcached 和 Redis 都有较好的的数据缓存性能，但是他们都属于系统额外的插件，增加了开发和维护成本。JDBC 缓存技术是在 JDBC（Java Database Connectivity，Java 数据库连接）基础上实现服务器缓存的一种方式，JDBC 是一种 Java API，用来实现对多种关系型数据的统一操作，应用广泛，目前大部分服务器应用都会通过 JDBC 来操作数据库。JDBC 缓存技术通过修改扩展 JDBC 的 API 接口，将满足条件的查询语句结果集前置到业务服务器中，下次遇到同样的请求，直接从服务器缓存中读取，无需重新与数据库建立连接，减少了资源消耗，缩短了响应时间。因为直接借助系统必不可少的 JDBC 技术，没有额外插件，比 Memcached 和 Redis 更加轻便。但是目前的 JDBC 缓存技术仅仅只是初步的设想和实现，没有应用到实际项目中，且对缓存的置换策略和一致性维护策略研究不足，整体性能还有待改进。

数据库缓存分为内存数据库技术和数据库缓存技术。内存数据库将工作数据集存放在内存中，数据的存取工作不再涉及磁盘的 I/O 操作[25]，效率更高，其对应的是磁盘数据库。现在常见的数据库如 MySQL，MongoDB 等，都是磁盘数据库。磁盘数据库需要频繁地访问磁盘，大量的 I/O 操作会降低数据库的性能。内存数据库因为直接将数据放在内存上，其读取速度比磁盘数据库要高出很多。典型的内存数据库有 SQLite、Altibase、Oracle Berkeley DB 等[26]。但是内存数据库给内存带来很大负担，且需要相应的备份、日志记录等保护机制保证数据安全。

另一种数据库缓存主要是指数据库的查询缓存[27]，利用数据库系统自身的机制，建立数据表高速缓存区，重复的请求直接读取缓存区内容回复，减少了查询时间，提升了查询效率。这种方式不能减小业务系统产生的增删改的压力，无法避免连接数据库的开销。

目前的三类缓存技术，浏览器缓存、服务器缓存和数据库缓存，浏览器缓存只能实现特定客户端的优化，无法多端多用户统一优化，且使用不当会带来客户端资源更新不及时的问题；数据库缓存仍然没有避免业务服务器与数据库建立连接的资源消耗；服务器缓存是一个很好的研究方向。并且本文的电动自行车智能充电系统中，有着大量的交互和数据操作，其重点在于服务器部分，服务器缓存符合本系统的需求。

## 1.3 研究内容及章节安排

本文设计实现了一套电动自行车智能充电系统，为用户提供远程充电和多种智慧服务，为管理者提供可视化数据管理。在实现系统功能的基础上，着重优化系统性能。设计基于四叉树网格划分的邻近充电桩快速搜索算法，提升了页面响应速度；设计改进的 JDBC 缓存机制，减小数据库的连接开销，提升了系统的并发能力。本文的章节安排如下：

第一章 绪论。首先分析了电动自行车充电桩发展现状及发展趋势，指出了当前充电服务系统存在的问题，明确了论文的研究意义。接着分别从电动自行车充电系统、空间查询、缓存三个方面，研究了国内外发展现状，为本文设计提供指导。

第二章 电动自行车智能充电系统设计及实现。本章首先明确系统设计需求，然后总结了系统架构，对总体功能和各模块进行了简要分析和介绍，接着详细说明各个子模块的内部机制和功能结构，最后展示相关实物及应用。

第三章 基于四叉树网格划分的邻近充电桩快速搜索算法。对用户服务的核心功能—邻近充电桩搜索进行优化。首先引入空间查询概念，研究已有空间查询技术，结合本文场景，提出一种基于四叉树网格划分的邻近充电桩快速搜索算法，设计 B-GeoHash 编码，对二维数据降维，并提取出前缀包含特性。在搜索时分为剪枝和过滤两个阶段，实现以用户为中心的特定范围内充电桩的快速搜索。接着，分析了算法复杂度，并对算法特性进行总结。最后，进行准确性和耗时性能测试，该算法在 30 万数据量时也可以 1s 内得到结果，性能良好。

第四章 基于 JDBC 的数据缓存研究及实现。首先由高并发问题引出服务器缓存，比较现有几种服务器缓存技术之后选择 JDBC 缓存技术作为研究方向，指出现有 JDBC 缓存技术的不足，并加以改进。通过修改扩展 JDBC 的接口，采用合理的数据结构，实现缓存存储及管理，并对缓存置换策略和数据一致性策略进行了研究设计。最后，实验测试该机制的命中率和响应时间性能，测试表明本文设计的 JDBC 缓存方案可以将系统的响应速度提高一倍

第五章 总结与展望。对全文的主要工作和研究内容进行归纳总结，对系统设计的不足之处提出改进方向，为后续研究提供指导。

# 第二章 电动自行车智能充电系统设计及实现

针对当前电动自行车充电桩市场存在的智能化程度不够、用户体验差等问题，本文设计了一套智能充电系统，涵盖硬件设计、云平台构建及客户端开发，实现了对充电桩的远程智能控制、可视化管理，同时服务用户和企业。

## 2.1 系统设计

### 2.1.1 需求分析

根据调研，当前已有联网式电动自行车充电系统设计存在硬件设计不足、配套软件服务落后、系统性能差等问题，在此基础上，对于需要设计的电动自行车智能充电系统，总结出以下几点需求：

（1）实现设备联网，实时上报充电数据并接收云端指令，实现远程控制。

（2）硬件设计保障充电安全和电池健康。充电安全方面，主控芯片程序需要采集实时充电数据，进行过流、过载、漏电判断，在异常情况下自动断电，保证安全。电池健康方面，做到充满自停。以往居民充电，因为无法判断电池充满所需时间，往往存在过充或充不满的情况，不利于电池健康，设计充满自停功能可以延长电池使用寿命。

（3）开发配套软件服务，让用户可以通过客户端实现便捷充电及个人充电数据管理。客户端设计应遵循人性化、操作简单、服务完善的原则，同时考虑不同职责的人员需求，不仅为充电用户提供服务，也要为企业管理员、运维人员等提供配套软件应用，实现数据价值最大化。

（4）整个系统应当稳定可靠，在数据量大、并发量大的情况下都有较好的性能，这样才能增强用户粘性，更好地推广产品。

### 2.1.2 整体框架

采用物联网技术，设计了电动自行车智能充电系统，整个系统可分为三部分：硬件系统、云平台、客户端，整体架构如图 2.1 所示。

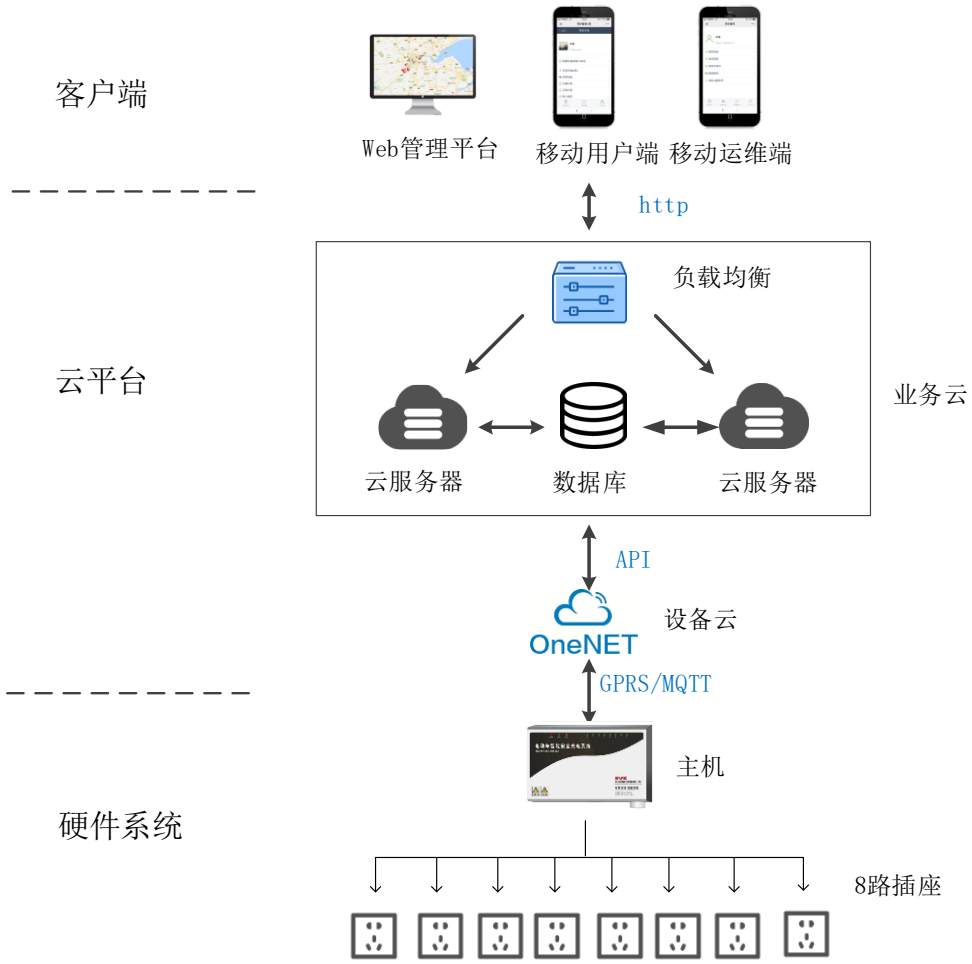



图2.1　电动自行车智能充电系统架构

硬件系统分为主控芯片、充电检测模块、继电器控制模块、电源模块、GPRS 通信模块五个模块。主控芯片与 GPRS 模块相连，接收到 GPRS 模块发来的充电命令后，主控芯片控制对应线路的继电器模块，打开插座开关，同时进行充电检测。GPRS 模块将收集到的充电数据上报到云平台。充电结束后，继电器模块控制开关关闭，GPRS 模块将状态变更推送到云平台。

云平台分为设备云和业务云两大模块。设备云与充电桩建立连接，实现充电数据采集、数据监控和指令下发；业务云负责具体业务服务接口，进行业务数据存储和管理，通过 RESTful API（Application Programming Interface，应用程序接口）对客户端软件提供调用功能。

客户端通过调用 API 来完成具体交互功能。Web 管理平台为企业管理者提供设备管理、可视化分析等功能；移动用户端实现扫码充电，实时推送充电信息，提供多样化智慧服务；移动运维端支持随时随地录入设备、测试通信，为管理者、运维人员提供便捷的设备管理、收益分析服务。

该系统可进行充电安全防护，提供过流、过载、漏电保护，实现充电异常自动断电、电池充满自停、充电器插头拔出自动断电。在客户端上充电状态实时显示，费用实时结算，充电完成及充电异常实时微信通知。用户可一键导航，快速找到邻近充电桩。企业管理员、运

维人员可通过软件应用进行高效的系统管理和远程运维。

## 2.2 硬件设计

充电桩硬件部分包含五个模块：主控芯片、充电检测模块、继电器模块、电源模块、GPRS 通信模块，结构如图 2.2 所示。

GPRS 通信模块发送控制信号，主控芯片接收到信号后执行对应指令。充电检测模块在充电过程中实时检测电流、功率等数据，数据传递给主控芯片，主程序进行过流、过载、掉电、浮充等状态判断。主控芯片通过继电器模块，实现插座通断控制。GPRS 模块接收云平台发送的指令，并将充电状态等数据上传。电源模块为控制电路各模块供电。

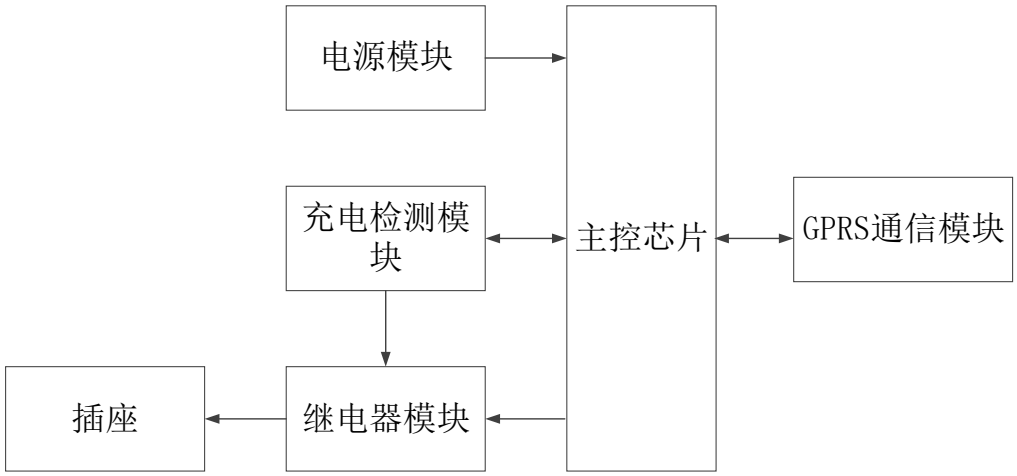



图2.2　充电桩硬件结构

### 2.2.1 MCU 设计

主控芯片是硬件系统的核心，原理图如图 2.3 所示。

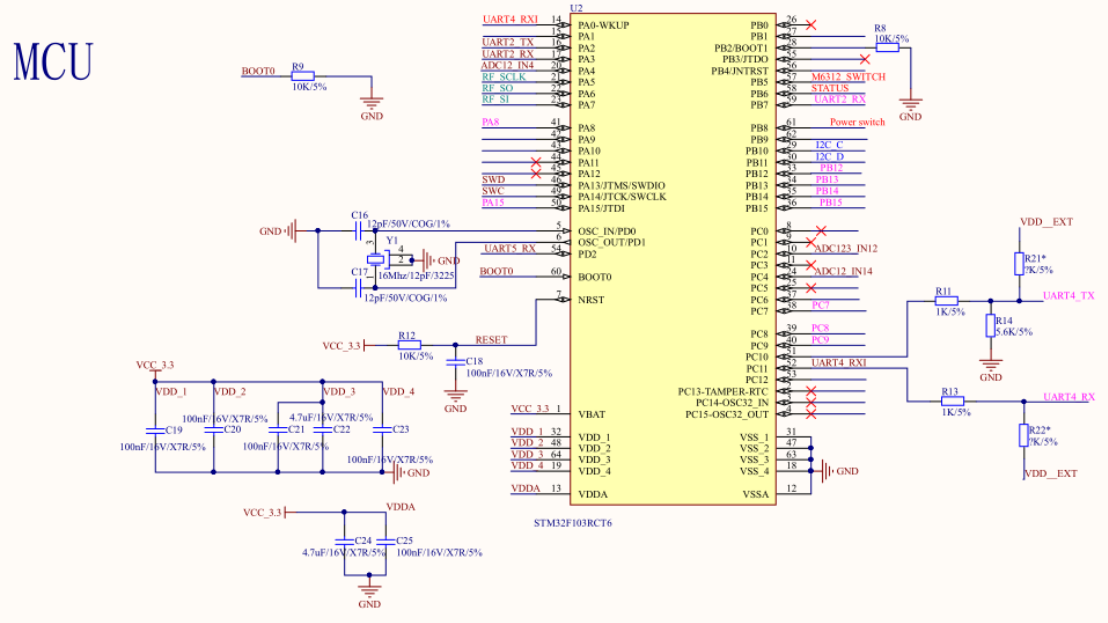



图2.3　MCU 原理图

主控板 MCU 选择 STM32F103RCT6，软件程序设计有过载检测、漏电检测、浮充等功能。通过处理检测到的电流、功率等的异常情况保障充电安全，只关注充电过程，与电池的参数或者容量无关。充电检测模块将实时采集的充电数据传给主控芯片，主控芯片根据功率及其变化进行相应动作，保护充电安全和电池健康，具体如下：

（1）浮充保护：本文系统设计有充满后浮充一小时的功能，对电池进行保养。当前市面上的电动自行车电池以铅酸蓄电池为主，如图 2.4 所示，充电分为三个阶段：恒流阶段、恒压阶段和浮充阶段。恒流阶段采用恒定电流充电，避免低电量时电流过大造成电池发热严重，引起火灾。恒压阶段采用恒定电压充电，保证电池达到额定电压。浮充阶段是铅酸蓄电池保养的关键。很多充电桩缺少这一阶段，往往检测到功率接近 0 的时候就认为电池充满，进行断电。但是浮充对蓄电池是很必要的，浮充用小电流继续充电，可以抑制活性物质硫酸盐化，延长电池使用寿命，补充电池自放电造成的容量损失。

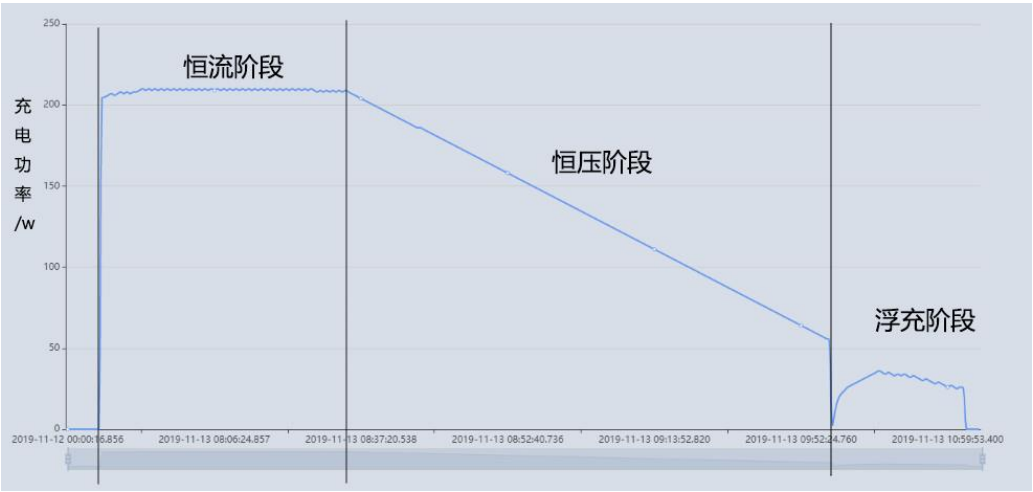



图2.4　铅酸电池三段式充电过程

（2）过充保护：充电时间过长，电池持续发热，容易引发火灾。主控芯片检测到电池充满且已完成 1 小时浮充时，会自动断电，避免长时间充电带来的安全隐患。另外，限制电池充电时长，当电池充电时间达到 10 小时，自动断电。

（3）过载保护：当一路插座中接入过多负载时，电流过大，会出现短路、过热起火等情况。本文设计的智能充电系统限制最大充电功率 800w，基本可以杜绝居民在插座上私拉电线接多个电池充电的行为。

插座的自动断电通过继电器模块实现，如图 2.5 所示，其本质是用一个小电流回路控制一个大电流回路的中断。当主控芯片检测到充电异常情况，对应引脚输出高电平，使闭合的开关断开，保护电路安全。

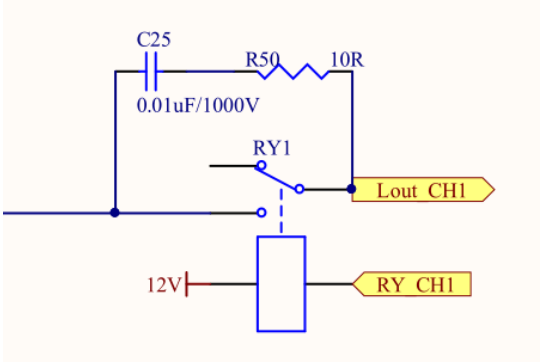



图2.5　继电器模块原理图

## 2.2.2 GPRS 通信模块设计

GPRS 模块实现云平台和充电桩主控板之间的通信。一方面，接收云平台的数据和指令，下发给具体的控制单元；另一方面，收集模块信息上报给云平台。原理图如图 2.6 所示。

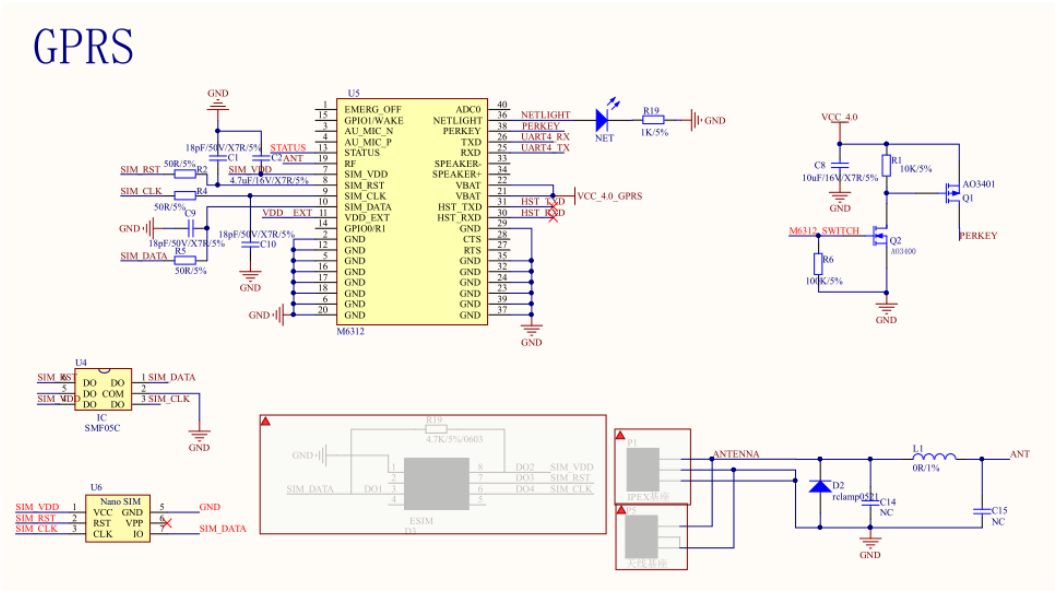



图2.6　GPRS 模块原理图

GPRS 模块负责与云平台保持长连接，维持数据通信；进行设备状态的实时查询和数据上报。图 2.7 描述了其内部处理机制，表 2.1 总结了其主要功能。

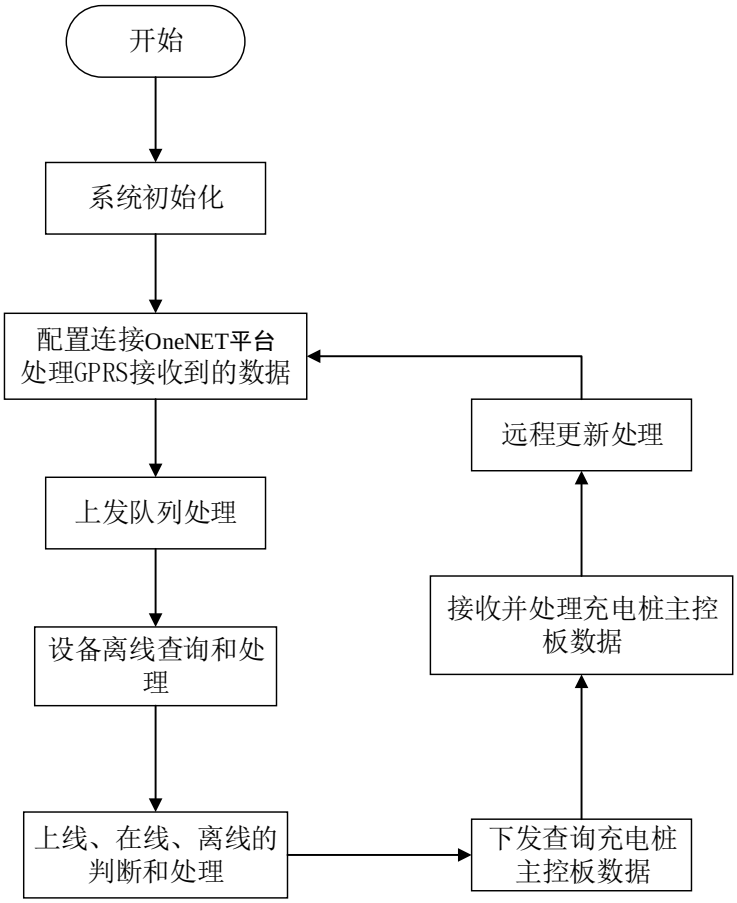



图2.7　GPRS 模块内部处理机制

表2.1　GPRS 通信模块功能表

| 序号 | 功能 | 技术细节 | 备注 |
|---|---|---|---|
| 1 | 地址识别 | 1 个 GPRS 模块下挂 8 个充电接口 | |
| 2 | 开关控制 | 单独控制每一个充电接口 | |
| 3 | 充电状态上传 | 统计充电接口是否在充电，采集充电信息 | |
| 4 | 电量统计 | 每路充电接口的实时电量信息上传，暂定 1 分钟上传一次 | |
| 5 | 浮充 | 判断浮充状态 | 浮充 1 小时 |
| 6 | 电池充满状态判断 | 判断是否充满，将充满信息上传 | |
| 7 | 充电时间设置 | 支持云端进行充电时间设置 | 时间到后断开电源 |
| 8 | 按键 | 复位/重置 | |
| 9 | 与充电桩通信 | 采用 I2C | |
| 10 | 异常信息上传 | 根据浮充条件和初始充电时间判断 | 如用户插头未插好 |
| 11 | 设备故障信息上传 | 1、 单个插座被使用 30 次，但是没有监 | |

| | | | |
|---|---|---|---|
| | | 测到功率变化<br>2、 单个插座 1 个月没人使用 | |

### 2.2.3 通信协议设计

充电桩主控板通过串口与 GRPS 模块通信，本文自主设计了通信协议，帧结构如表 2.2 所示。帧头 H 标识一帧信息的开始，其值为 0x68。控制码 C 包含操作行为和操作对象。数据域 DATA 根据控制码的不同而变化。校验码 CS 用来防止丢包和错误帧，取第一个帧起始符开始到校验码之前的各字节的二进制算术和，不计超过 256 的溢出值。帧尾 E 标识一帧信息的结束，其值为 0x16。

表2.2 自定义帧结构

| 帧头 H | 控制码 C | 数据域 DATA | 校验码 CS | 帧尾 E |
|---|---|---|---|---|

控制码 C 的格式如表 2.3 所示，接收方进行数据回复时，不修改控制码，原样返回。

表2.3 控制码 C 格式

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 备注 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 地址 | | | | | | 读取插座数据 |
| 0 | 1 | 0 0 0 0 0 0 | | | | | | 读取重启次数 |
| 1 | 0 | 地址 | | | | | | 关闭插座 |
| 1 | 1 | 地址 | | | | | | 开启插座 |

D7：0 表示读取，1 表示设置。

D6：读取数据时，0 表示读取插座数据，1 表示读取充电桩重启次数（地址内容为 0）。设置数据时，0 表示关闭插座，1 表示开启插座。

D5～D0：表示插座的编号。

数据通信采用主从结构，GPRS 模块作为主机，充电桩控制板作为从机，数据的读写由主机发起，从机回复。

#### 2.2.3.1 开启插座

开启插座时，GPRS 模块发送开启插座帧，数据域中的数据为充电时间，以分钟为单位，充电桩接收到数据，执行开启操作后，再回复同样的帧。

示例：

表2.4 开启插座帧示例

| | **H** | **C** | **DATA** | | **CS** | **E** |
|---|---|---|---|---|---|---|
| 发送 | 68 | C0 | 02 | 58 | 28 | 16 |
| 回复 | 68 | C0 | 02 | 58 | 28 | 16 |

表 2.4 表示，GPRS 模块发送开启插座 1 的帧，持续时间为 600 分钟，充电桩执行开启操作后，进行回复。

#### 2.2.3.2 读取插座数据

读取插座数据时，充电桩主控板回复帧的数据域（DATA）内容如下：

表2.5　主控板回复帧的数据域

| 插座状态 | 功率高位 | 功率低位 | 剩余充电时间高位 | 剩余充电时间低位 |
|---|---|---|---|---|
| 1byte | 1byte | 1byte | 1byte | 1byte |

总共有 5 个 byte，第一个 byte 表示当前插座的状态，具体含义如表 2.6 所示。

表2.6　插座状态位设计

| bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | 保留 | 保留 | 保留 | 过流 | 电池故障 | 漏电 | 浮充 | 开关状态 |
| 0 | 默认为 0 | | | 否 | 否 | 否 | 否 | 关 |
| 1 | | | | 是 | 是 | 是 | 是 | 开 |

第二个和第三个 byte 表示功率（高位在前），如果该插座没有在充电，则以 0 进行填充。第四个和第五个 byte 表示当前插座剩余的充电时间（高位在前），如果该插座没有在充电，则以 0 进行填充。

示例：

表2.7　读取插座数据帧示例

| | **H** | **C** | **DATA** | | | | | **CS** | **E** |
|---|---|---|---|---|---|---|---|---|---|
| 发送 | 68 | 00 | NULL | | | | | 68 | 16 |
| 回复 | 68 | 00 | 01 | 02 | 1E | 00 | 78 | 89 | 16 |

表 2.7 表示，GPRS 模块发送数据，查询插座 1 的状态。充电桩回复，告知此时插座 1 处于充电状态，充电功率为 542W，剩余充电时间为 120 分钟。

#### 2.2.3.3 关闭插座

关闭插座时，GPRS 模块发送关闭插座帧，充电桩接收到数据，执行关闭操作后，再回复同样的帧。

示例：

表2.8　关闭插座数据帧示例

| | **H** | **C** | **DATA** | **CS** | **E** |
|---|---|---|---|---|---|
| 发送 | 68 | 80 | NULL | E8 | 16 |
| 回复 | 68 | 80 | NULL | E8 | 16 |

表 2.8 表示，GPRS 模块发送关闭插座 1 的帧，充电桩执行相关操作后，进行回复。

## 2.3 云平台设计

云平台分为设备云和业务云两部分，设备云收集管理设备数据，进行远程控制；业务云提供逻辑处理和业务数据存储服务，为应用软件赋能，业务处理程序中有两个比较重要的机

制：接口调用安全认证机制和微信登录认证机制。业务云通过设备云提供的 API 实现两者之间的交互。

### 2.3.1 设备云

设备云选用中国移动 OneNET 云平台，进行设备通信、设备数据存储与管理。OneNET 是中国移动公司开发的物联网公有云平台[28]，可以解决数据存储、设备管理、规则引擎、事件告警、协议适配等物联网开发共性问题，符合本系统的开发需求，并且有庞大的用户群体，经受过海量接入的考验，较为成熟。

GPRS 模块将采集到的数据通过 MQTT 协议上报到 OneNET 设备云平台，OneNET 对数据进行存储和管理，同时对业务云提供调用接口，使开发者能够利用 API 实现对设备云数据的读取和远程设备控制。平台的管理页面如图 2.8 所示。开发者可以根据具体需求进行数据流、触发器设置，监控管理接入设备。

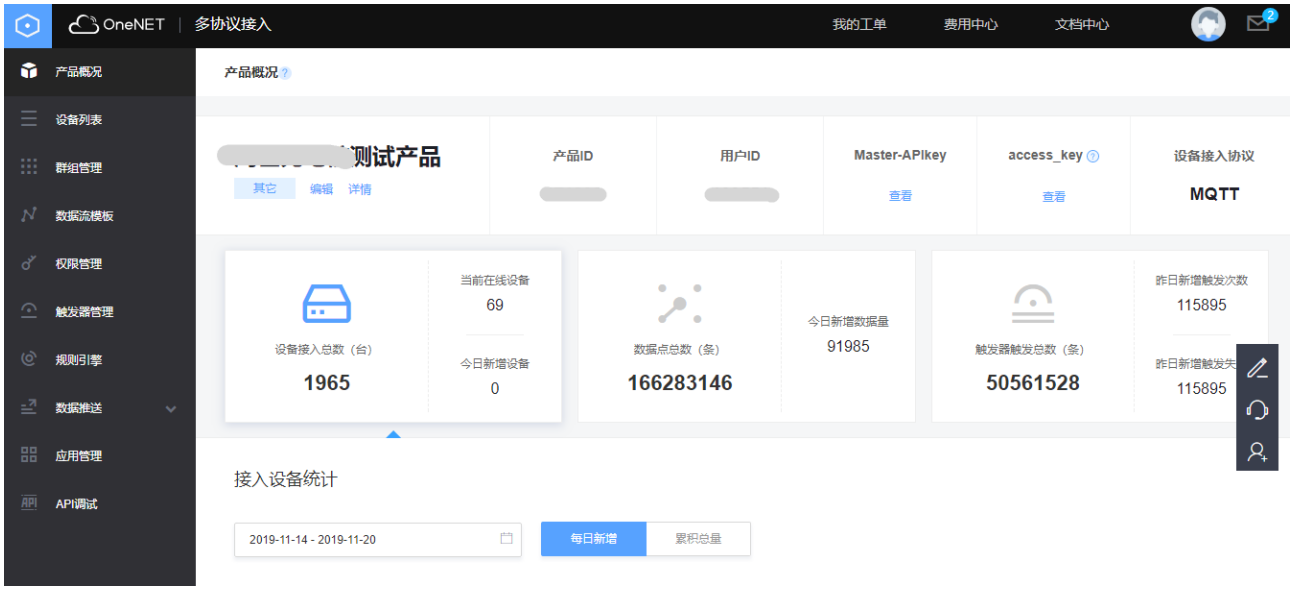



图2.8　OneNET 设备云管理平台

### 2.3.2 业务云

业务云由负载均衡、云服务器和数据库构成。采用两台阿里云服务器组成业务服务器集群，负载均衡对前端请求进行转发分配，缓解单台服务器压力；云服务器进行业务逻辑处理，为应用软件提供可靠功能接口；使用 MySQL 数据库进行数据存储。业务云整体架构如图 2.9 所示。

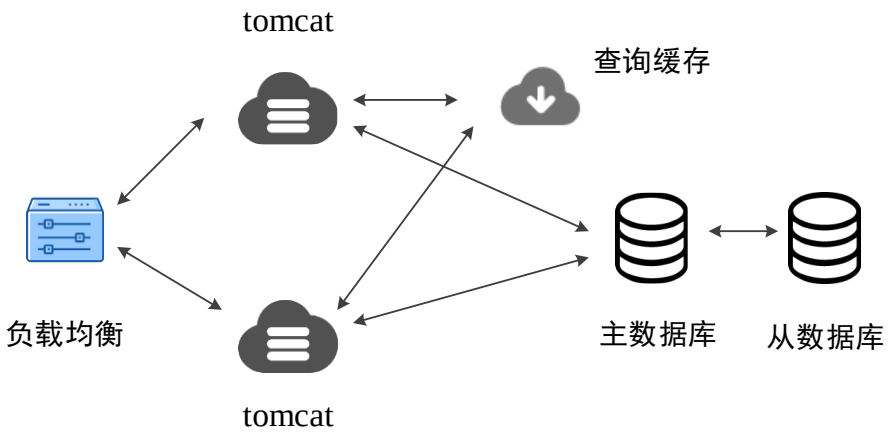



图2.9　业务云架构

#### 2.3.2.1 业务云技术架构

负载均衡采用阿里云高可靠弹性负载均衡，负载算法采用最小连接数轮询。数据库使用阿里云提供的 MySQL 云数据库，采用主备存储的方案，实现高可用。

后端程序采用 SpingBoot+Spring+JDBC 技术栈进行开发，运行在 tomcat 上。tomcat 监听特定端口的 HTTP 请求，按照 servlet 规范包装后将其转发到应用程序中[29]。包装后的 HTTP 请求首先转发到 Sping MVC，Spring MVC 根据请求的 URL、Method 等信息对应到具体的 API，然后处理业务程序，处理完成后返回响应结果。若在业务处理中操作数据库，则通过 JDBC 技术来实现[30]。

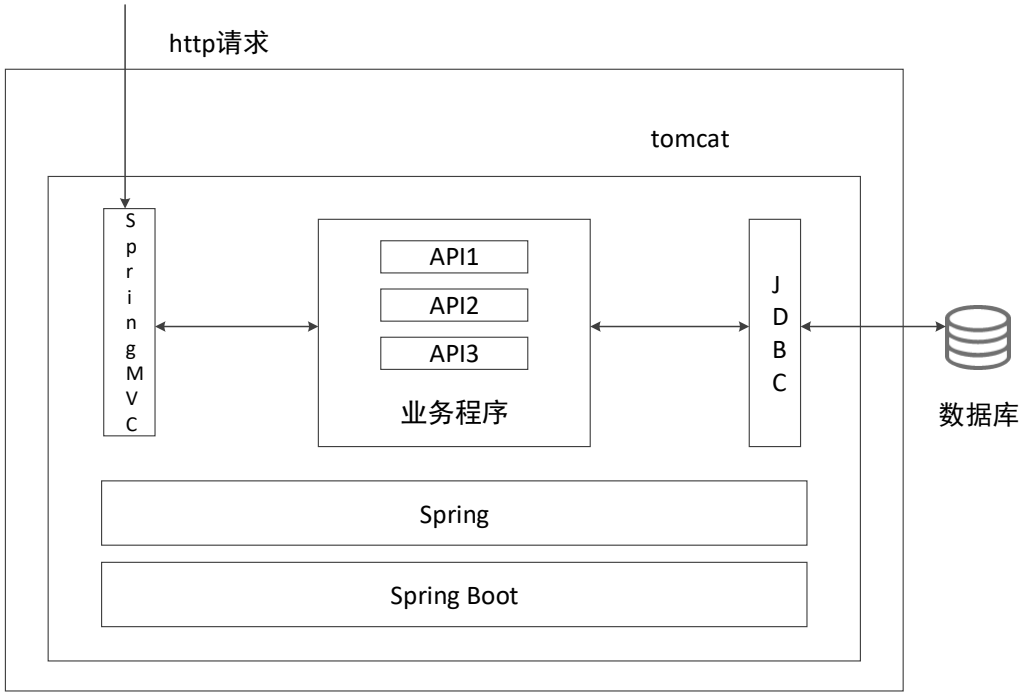



图2.10　后端技术架构

最后，采用 Docker 容器部署业务程序。Docker 通过定制应用镜像可以实现部署、持续集

成、持续交付[31]，有利于业务层系统功能与性能的扩展。

#### 2.3.2.2 接口调用安全认证机制

HTTP 请求是无状态的，服务器中不保存客户端的状态[32]，每次请求客户端需要重新带上身份信息。同时我们需要对请求来源进行安全认证，保证请求来自可靠的发起者。本文采用 JWT（JSON WEB TOKEN）机制进行 API 无状态安全校验。JWT 是一种 JSON 格式的 token 鉴权方案，加密时可以存储相关信息，解密后可以获得所需数据。采用 JWT 机制生成的 token 总共有三部分： header，payload，signature[33]。header 中记录了加密算法，payload 存储过期时间及自定义信息，signature 存储对 header 和 payload 的签名，防止 token 被篡改。

本文设计的 token 验证机制流程是：如果用户登录成功，将用户唯一标识 userId 作为参数生成 token，将 userId 和 token 返回给请求者。接下来网站发起的所有 API 请求头都要带上 userId 以及 token，服务器对 token 解密后可以得到里面包含的 userId 和过期时间信息。如果 token 未过期，和传入的 userId 比较，结果相同即认证通过，然后才能进行具体业务操作。用户登录和注册这类登录前级的 API 不进行 token 认证。流程图如下：

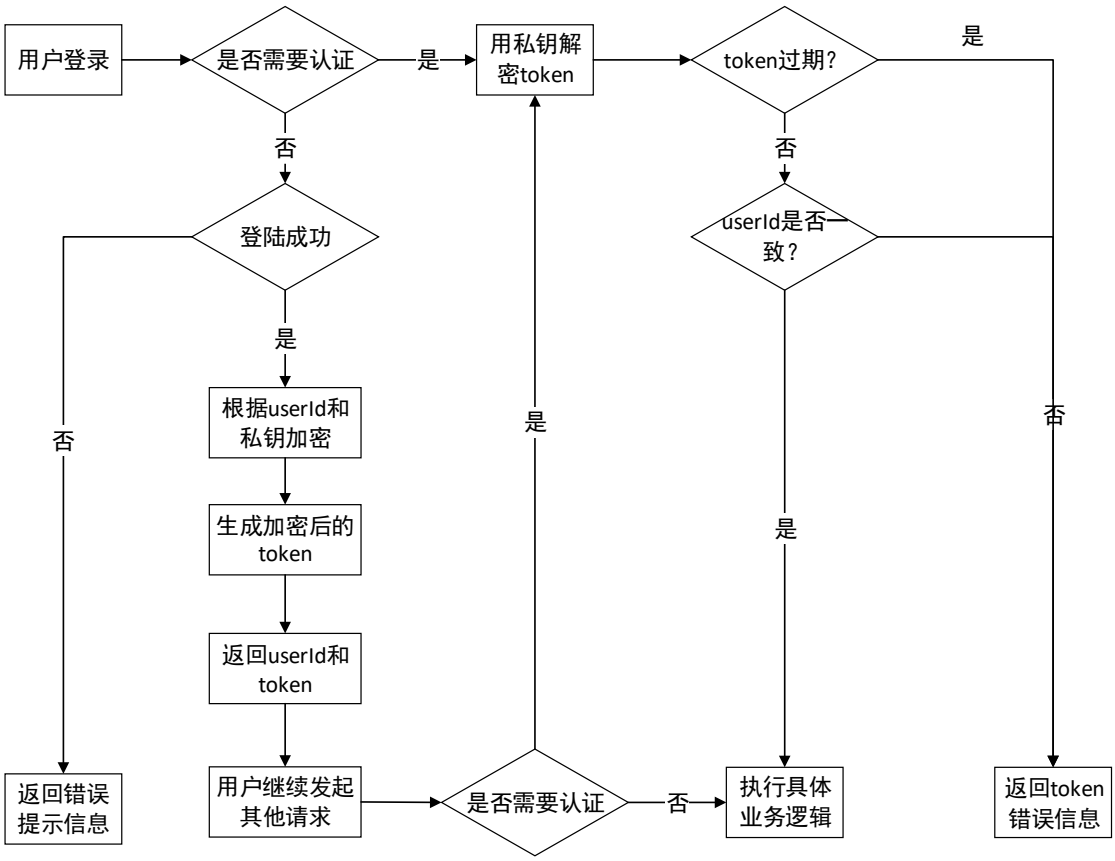



图2.11　API 认证流程

#### 2.3.2.3 微信登录认证机制

本文基于微信网页生态开发移动端应用，借助微信本身的优势，我们可以实现用户静默

登录。只要将用户微信与系统中的唯一身份识别信息对应起来，以后用户只要打开系统就可使用，不需要重复登录。

用户访问公众号菜单，到获取网页内容前，需要经过一系列认证操作，如图 2.12 所示，具体为：

（1）微信浏览器向业务服务器发起授权地址获取请求，业务服务器生成 sessionId。然后将其与授权地址一起返回。

（2）微信浏览器访问授权地址后，弹出用户授权界面，用户点击允许获取信息。

（3）用户允许后，再由微信服务器携带关键参数重定向来访问业务服务器。

（4）业务服务器根据传入的关键参数来获取用户的 OpenID 和 token；并将其与 sessionId 关联，其中 OpenID 是用户在当前公众号下的身份标识[34]，sessionId 在前后端交互过程中实现会话保持。

（5）最后将页面重定向至目标系统主页，并将 sessionId 存储在 cookie 中，用户再次进入时，系统直接读取 cookie 记录识别用户身份。

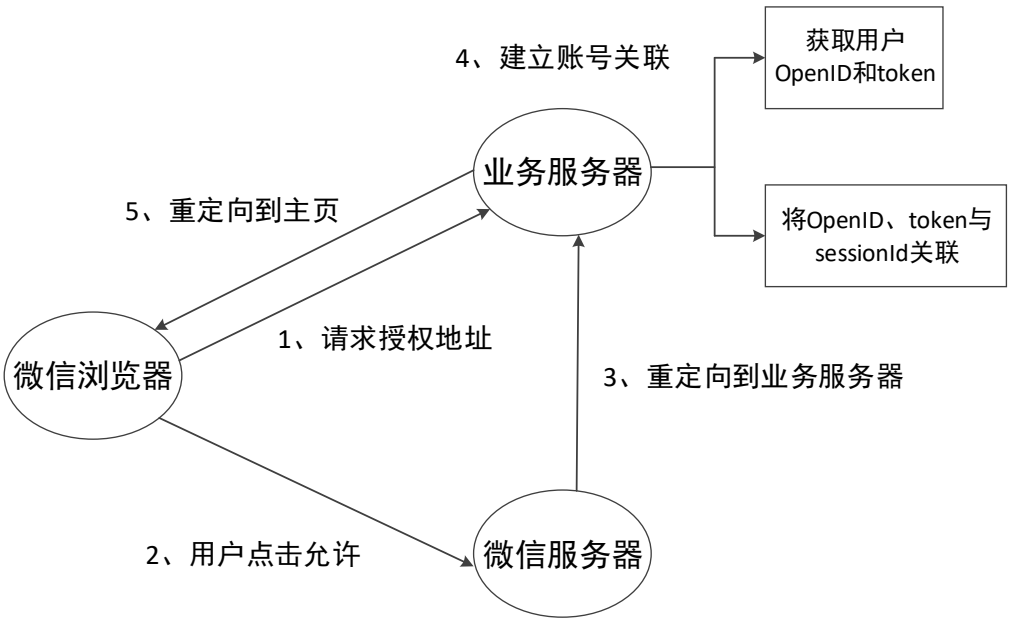



图2.12　微信认证流程

## 2.4 客户端设计

整个充电桩系统会涉及不同职责的人员，主要分为：（1）企业管理者：开发充电桩系统的企业人员，他们可以选择自己经营充电桩或者将充电桩卖给客户。（2）加盟商：即企业售卖充电桩及相关服务的客户，加盟商从企业处购买充电桩进行自我经营。（3）运维人员：充电桩的安装和维护人员。（4）用户：使用充电服务的用户。

企业管理者对所有充电桩、客户、用户都有权限，加盟商只对自己拥有的充电桩有权限，运维人员只负责与运维相关的事务，用户仅仅是系统的使用者。

根据他们的角色特性，将客户端开发分为三个系统：Web 管理平台、移动运维端和移动用户端。

移动用户端给充电用户使用，提供扫码充电、地图服务、充电管理等功能。移动运维端面向企业管理者、加盟商和运维人员，提供设备录入、数据统计、收益分成等功能。Web 端开发管理平台，为企业管理员提供设备管理、可视化服务和数据分析等服务。

### 2.4.1 客户端技术架构

Web 管理平台基于 PC 浏览器开发，移动端我们选择微信网页作为载体。

从智能手机兴起到现在，手机 APP 市场呈爆炸式发展状态，各种各样的 APP 让人目不暇接。最开始的新奇感过后，人们逐渐感到疲惫。面对 APP 轰炸，人们更倾向选择更小更轻便的实现，这也是近年来多端小程序、快应用流行的原因。尽管现在手机内存不断增加，但很多人还是会觉得不够用，此时再让他们装载新的 APP 是不友好的。在这样的情况下，开发微信公众号网页这一方式，具有很大优势，同时它还有其他优点，总结如下：

（1）微信是每个人都会用到的软件，基于微信生态开发，不用额外安装应用软件，用户使用起来没有负担。

（2）基于微信开发可以实现多端复用，无需为安卓和苹果用户开发两套代码。

（3）利用微信支付，可以轻松保障资金和交易安全。

（4）开发的系统与公众号关联，可借助公众号进行消息推送，更好地服务用户。

（5）收益分成时，可以实现定期自动将资金支付到对方微信零钱中，快捷方便。

开发完成的微信网页通过链接关联到公众号，通过公众号的菜单即可进入系统。进入方式如图 2.13 所示。


图2.13　微信网页入口

微信网页开发仍然是基于微信内置浏览器，所以移动端和 Web 端的设计都是基于浏览器的。整个充电桩系统的软件部分采用 B/S（Browser/Server，浏览器/服务器）模式进行前后端分离开发。后端不用再负责模板渲染输出工作，只需专注自身 API 开发和程序逻辑处理。前端通过 HTTP 协议调用后端封装的 API 接口，进行功能交互和数据传递。前端页面的逻辑和渲染由前端程序独立负责，且前端基于 MVVM（Model-View-ViewModel）模型开发[36]，视图和数据模型低耦合，开发更灵活。

前端采用 Vue.js 框架进行开发。Vue.js 框架是目前前端开发最流行的框架之一[37]，开发者直接使用封装好的组件模板，简单易上手。Vue.js 是一个渐进式的框架，其核心是声明式渲染和组件系统[38]，路由、状态管理、构建工具等解决方案相互独立，可以使用官方工具也可以在核心的基础上任意选用其他部件。框架做分层设计，每层都可选，不同层可以灵活接入其他方案。

Vue.js 框架开发的项目是一个单页面应用（single page web application，SPA）[39]，即整个系统只有一个主页面，其他页面都是包含在主页面中的，跳转只刷新局部资源，不会向服务器发送请求，页面切换流畅用户体验好，减轻了服务器的压力。

前端技术架构如图 2.14 所示，大致可分为组件系统、路由管理、状态管理、网络请求、第三方组件五部分。组件系统是核心部分，包含前端的页面资源；路由系统管理单页面应用的路由切换、路由状态；状态管理负责全局状态的维护；网络请求实现前后端的通信及交互；第三方组件帮助我们更好地开发系统，如使用 BMap、ECharts 组件进行可视化开发。vue-cli、npm 、Webpack 是一些构建工具，帮助进行工程搭建、插件安装、打包压缩。

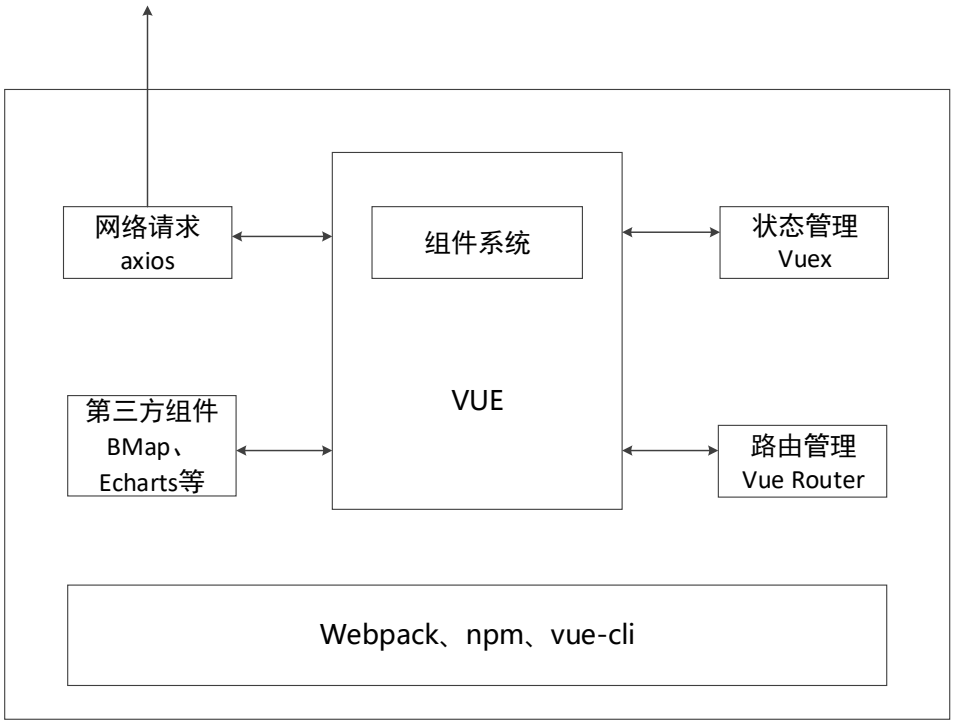



图2.14　前端技术架构

### 2.4.2 移动用户端

移动用户端面向有充电需求的用户。以服务用户为设计宗旨，开发了扫码充电、远程控制、个人充电管理等功能，系统的详细功能如表 2.9 所示。

表2.9　移动用户端功能表

| 一级分类 | 二级分类 | 功能概述 |
| --- | --- | --- |
| 个人中心 | 信息栏 | 显示用户微信昵称、头像、手机号信息 |
| | 我要充值 | 充值服务 |
| | 正在充电 | 用户当前充电信息，如功率、计费等 |
| | 常用电站 | 用户最常用的 5 个充电桩 |
| | 充电记录 | 用户充电记录 |
| | 充值记录 | 用户充值记录 |
| | 家人模式 | 通过输入插座序列号和序号，实现对插座的远程控制 |
| 扫描插座 | | 系统内置扫码功能，直接扫描插座二维码充电 |
| 附近电站 | 地图服务 | 定位当前位置，显示附近 10km 范围内的充电桩 |
| | 设备列表 | 展示地图上的充电桩详情，点击对应项可以选择充电或导航 |

用户版中，充电页面是一个核心部分，如图 2.15 所示，页面包含以下几个方面：（1）充电插座信息，包括插座位置、充电时间、消费金额、使用状态等；（2）开始充电、停止充电按钮，直接控制插座的开关状态；（3）其它，功率区间价格说明、账户余额、充值页面跳转等。

系统为用户提供了两种充电模式：充满自停和定时充电。充满状态根据充电功率进行判断，系统自动监测，保障电池健康。定时模式可以让用户进行自我时间规划，灵活多变。

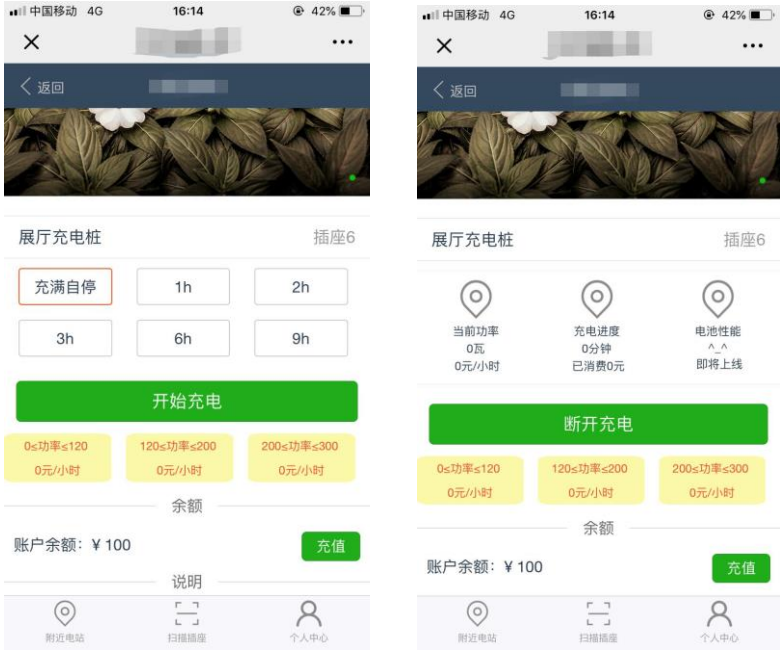



图2.15　移动用户端充电页面

用户通过扫码或其它方式进入充电页面，首先要获取当前插座的所属充电桩、使用状态等信息，并根据这些信息对页面中的按钮进行对应设置，如当前插座被其他用户占用，充电按钮显示“正在使用中”，不可点击；如当前插座被当前用户使用，充电按钮显示“断开充电”，允许点击，进行断电操作；如插座空闲，充电按钮显示“开始充电”，允许点击，进行充电操作。

开始充电前，要判断用户是否满足充电条件，主要对是否绑定手机号和是否有余额进行判断，如不满足条件则提示用户进行相应操作。开始充电后，采用轮询的方式定时更新页面中的充电数据。其中刚开始充电时定时器间隔时间短是为了防止用户的插头未插好，系统可以及时检测到，作出相应提示。整体流程如下：

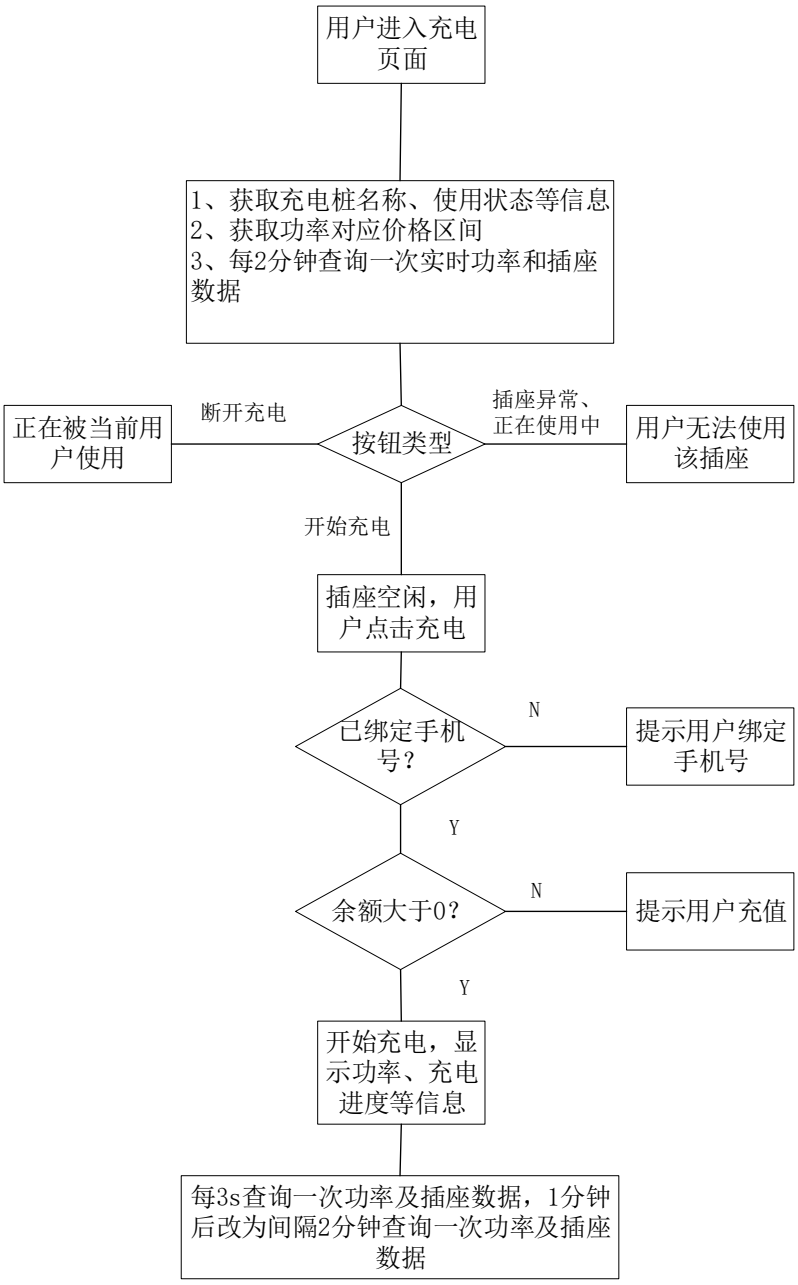



图2.16　充电页面程序流程图

充电结束后，公众号会推送相关信息，如图 2.17 所示。


图2.17　公众号消息推送

### 2.4.3 移动运维端

移动运维端面向企业管理者、加盟商和运维人员。在实地安装充电桩时，需要进行设备录入、通信测试等操作，设计具有这些功能的软件可以帮助运维人员简化工作，避免人工记录信息造成的差错。但是，场地中往往没有有线网络或者 Wi-Fi，用 PC 端 Web 系统是不可行的。本系统设计开发了微信端网页系统，帮助运维人员更好地进行设备安装和管理。同时为了让管理员、加盟商可以实时查看设备状况、营收情况，增加了设备管理、收益统计分析等模块。具体功能如表 2.10 所示。

表2.10　移动运维端功能表

| 一级分类 | 二级分类 | 功能概述 |
|---|---|---|
| 登录 | | 登录、注册、找回密码 |
| 个人中心 | 我的信息 | 个人信息管理 |
| | 修改密码 | 密码修改 |
| | 修改手机号 | 手机号修改 |
| | 微信绑定 | 绑定当前微信与手机号，用于收益分成，直接将收益支付到对方微信零钱中 |

| 一级分类 | 二级分类 | 功能概述 |
| --- | --- | --- |
| 设备录入 | | 分三个步骤：信息录入、绑定插座、通信测试，经过录入，设备就可以正常联网使用 |
| 设备列表 | 设备信息查看 | 设备基本信息 |
| | 设备通信测试 | 设备通信功能测试 |
| | 价格-功率区间设置 | 设置功率区间对应的价格 |
| | 报警联系人 | 设置当前充电桩的报警联系人 |
| | 受益人 | 设置当前充电桩的收益分成比例及受益人 |
| | 移交设备 | 移交设备给其他人员 |
| | 解绑设备 | 解绑与设备的绑定关系 |
| 数据概况 | | 显示用户昨日收益、近一周收益、近一个月收益；用户自定义查询时间和充电桩，个性化查询 |

设备录入是移动运维端的基本功能，分为信息录入、绑定插座、通信测试三个步骤。首先录入充电桩的基本信息，然后将充电桩与插座进行绑定，形成对应关系，最后通信测试，检验是否安装成功，如图 2.18 所示。


图2.18　移动运维端设备录入流程

运维版的一个核心功能是收益分成。当前设计的分成模式是，先将所有资金汇总到企业账户，然后企业再根据客户设置情况进行结算分配。本文设计自动结算模式，在设备管理的第三方受益人中，客户自定义企业、自己、第三方受益人（如物业）对该充电桩的收益分成比例，后台每天凌晨进行自动结算，借助微信支付平台，将分成直接支付到受益人的微信零钱中，交易明细清晰，避免了人工计算的繁琐和误差。


图2.19　受益人设置

### 2.4.4 Web 管理平台

Web 后台管理平台是企业管理员用来对所有充电桩及客户、用户等进行统一管理的平台。设计设备信息、数据概况、设备管理、视频监控、客户管理、用户管理、报警信息、资金结算、系统设置九大模块，实现设备远程运维、业务数据管理、可视化分析等功能，功能概况如表 2.11 所示。

表2.11　Web 管理平台功能表

| 一级分类 | 二级分类 | 功能概述 |
|---|---|---|
| 登录 | | 登录、注册、找回密码 |
| 设备信息 | | 地图标识充电桩位置、在线情况等信息；点击具体充电桩显示详情； |
| 数据概况 | 设备概况 | 图表展示设备情况，包括每日激活、每日在线、累计激活、城市分布 |
| | 用户概况 | 图表展示用户情况，包括每日新增、每日活跃、累计用户 |
| | 客户概况 | 图表展示客户情况，包括每日新增、累计客户、城市分布 |
| | 资金概况 | 图表展示资金情况，包括每日收益、累计收益 |
| 设备管理 | 充电桩列表 | 列表展示所有充电桩，显示 SN、deviceId、所属客户等信息，提供插座二维码下载、远程控制等功能 |
| | 充电曲线 | 展示插座的实时功率采集曲线 |
| | 统计分析 | 提供设备城市分布、区域特性等分析图表 |
| 视频监控 | 视频列表 | 列表展示所有视频设备信息 |
| | 实时监控 | 对应视频监控设备的实时画面 |

| 一级分类 | 二级分类 | 功能概述 |
| --- | --- | --- |
| | 账号配置 | 视频设备 AppKey、Secret 的配置 |
| 客户管理 | 客户列表 | 列表记录所有客户（加盟商）及拥有设备信息 |
| | 客户分析 | 提供客户分布、收入等分析图表 |
| 用户管理 | 用户列表 | 列表记录所有充电桩用户基本信息、充值记录、充电记录等。 |
| | 用户分析 | 提供用户充值区间的分析图表，为促销充值活动提供参考 |
| 报警信息 | 当前报警 | 所有设备的当前报警信息 |
| | 历史报警 | 所有设备的历史报警信息 |
| 资金结算 | | 资金管理、收益分成等 |
| 系统设置 | 个人中心 | 当前系统用户信息及操作 |
| | 账号管理 | 高级权限的管理员对系统使用者的权限分配及管理 |
| | 报警联系人 | 全局报警联系人管理 |
| | 流量监控 | 管理 SIM 流量报警阈值 |

Web 系统中设计了具有特色的可视化管理模块，将所有充电桩展示在地图中，管理员对充电桩的位置、在线状态、详情等一目了然，信息面板中可实现设备管理、报警信息的快速跳转。

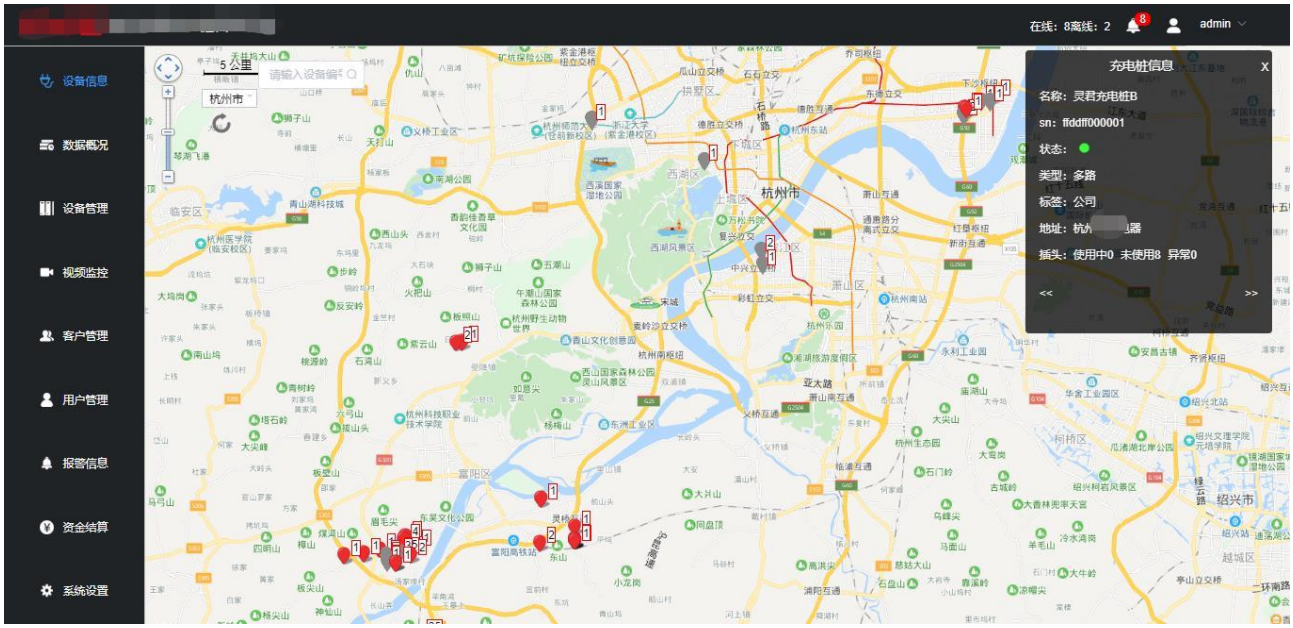



图2.20　Web 后台可视化管理

GPRS 模块采集设备的实时功率数据，这些数据构成了完整的充电周期，系统用折线图的方式对充电功率数据进行可视化展示，供管理员分析。图 2.21 展示了一个完整充电周期中的功率曲线。


图2.21　充电功率曲线

作为一个综合的管理平台，会有很多不同的角色，对应不同的资源权限。传统的权限控制方案是前端根据用户登录后后端返回的用户角色属性 usrRole 进行视图和操作的权限控制。但是对于单页面应用，因为资源在初始化的时候已经全部加载，即使页面中没有入口，恶意用户也可以通过直接输入目标地址的方式进入自己没有权限的部分。

本文设计了一种较为安全的权限控制方案。在初始化的时候只挂载 404 页面和登录页的路由，其他页面路由暂不生成。用户登录成功，后台再返回该用户拥有的权限资源 permission，前端处理后用 addRoutes 方法动态挂载路由，并有针对性的渲染视图。为了避免不安全问题，前端不存储 permission 信息，只存储用户的 userId，当用户刷新页面导致系统初始化时，根据 userId 请求 permission 重新渲染。整个流程如下。

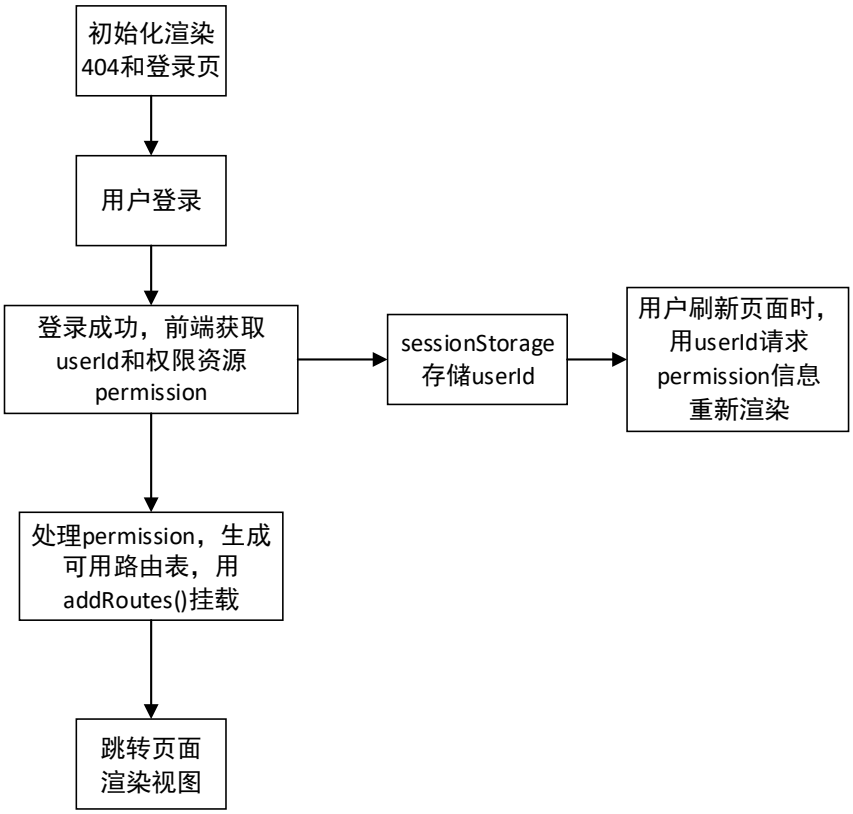



图2.22　Web 管理平台权限控制方案

### 2.4.5 页面响应速度优化

页面响应速度是影响用户体验的重要因素，本文设计的客户端系统都是单页面应用，整个系统只有一个主页面，其他页面都是主页面的子页面。用户进入系统，页面初始化，会加载所有页面资源，虽然系统内切换流畅，但是会牺牲首屏加载时间，造成用户刚进入系统的时候页面反应慢。客户端设计中采取了一些措施来解决这一问题，具体如下：

（1）通过在静态资源服务器上配置 gzip 压缩，减小打包后的工程资源大小，缩短响应时间。使用 gzip 压缩前的系统完成加载需要 8-10 秒，使用 gzip 压缩后只需要 2-3 秒。

（2）使用字体图标库代替图标图片，通过减少 http 请求来提高页面响应速度。每一张图片都是一个需要请求的资源，如果图片很多需要多次请求。字体图标库中所有字体图标的引用及配置写在同一 CSS 和 JavaScript 文件中，工程打包时与其他同类型的文件一起打包生成资源块，前端页面请求的时候只需一次请求就可以获取到所有的字体图标，对比图片图标减少了很多 http 请求。

（3）配置路由组件懒加载。单页面应用会在页面初始化的时候加载所有页面的路由资源，配置 Webpack 对代码进行分块打包，初始化只加载当前页面的资源，其他页面需要的时候再加载对应资源块。对于组件较多的系统，懒加载可以大大提升首屏加载速度。

## 2.5 系统展示

### 2.5.1 硬件展示

图 2.23 展示了 GPRS 模块 3D 图，图 2.24 展示了充电桩主板实物图。

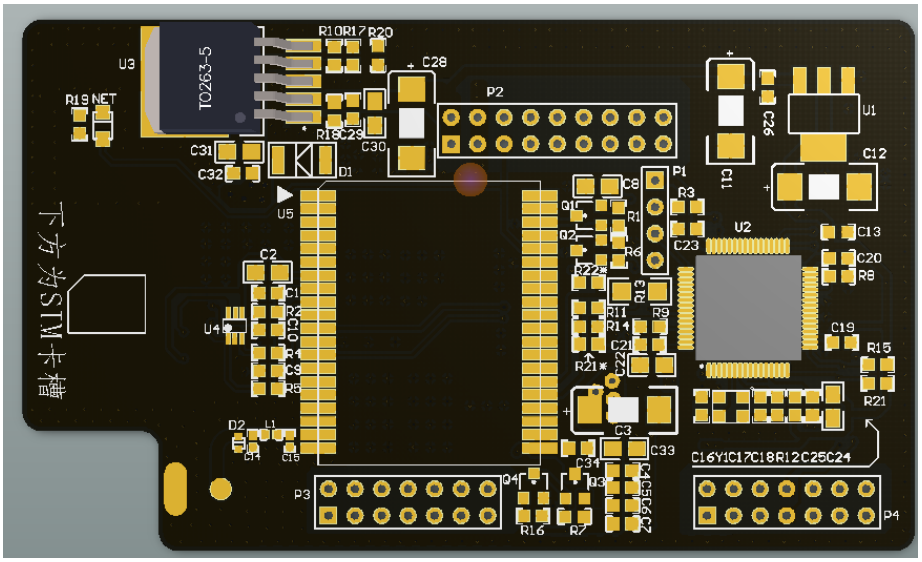



图2.23　GPRS 模块 3D 图

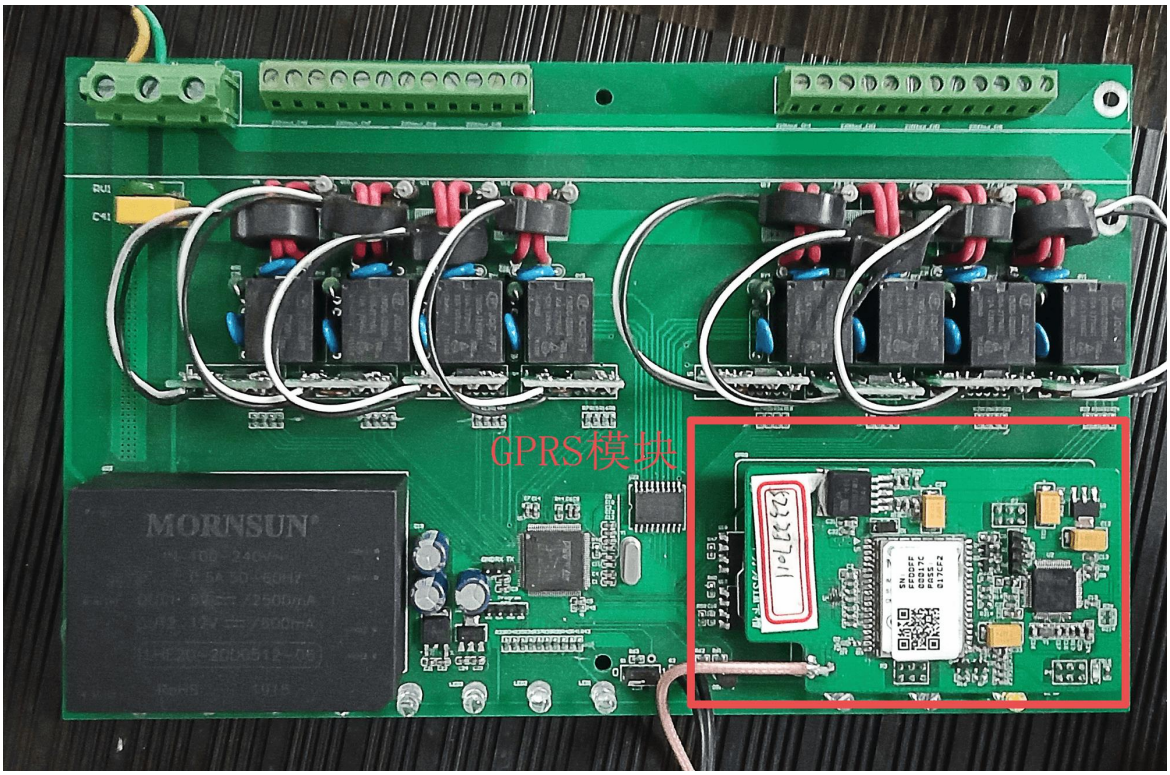



图2.24　硬件主板实物图

### 2.5.2 数据采集

将一辆装有 48V 12Ah 蓄电池的电动自行车接在插座 1 上，通过手机操作开始充电。

充电桩上的 GPRS 模块采集到充电功率数据，先上报到 OneNET 设备云平台，在 OneNET 平台上查看对应设备的数据流，如图 2.25，可以看到充电功率数据正常上报，OneNET 平台进行存储。

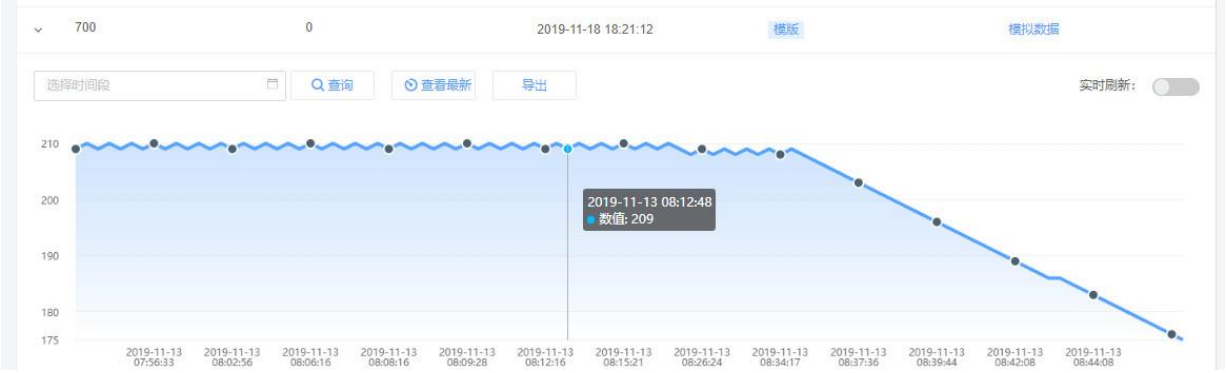



图2.25　OneNET 平台数据记录

同时，在我们开发的 Web 管理后台，也可以获取到实时的充电功率数据。


图2.26　Web 管理平台功率曲线

### 2.5.3 多种充电方式

本文设计的智能充电桩系统，充分为用户考虑，提供了多达 5 种充电方式。

（1）微信扫码充电，使用微信的扫一扫功能扫描插座二维码即可进入充电页面。

（2）系统内自带的扫码充电功能，点击“扫描插座”扫描插座二维码即可进入充电页面。


图2.27　充电方式-系统内部进入

（3）常用电站：在个人中心-常用电站中，记录用户最常用的 5 个充电站，用户可以在列表中选择对应充电桩的对应插座直接充电。


图2.28　充电方式-常用电站

（4）附近电站：附近电站为用户提供邻近充电桩搜索功能，用户可点击地图下方的充电站列表，选择对应插座进行充电。


图2.29　充电方式-附近电站

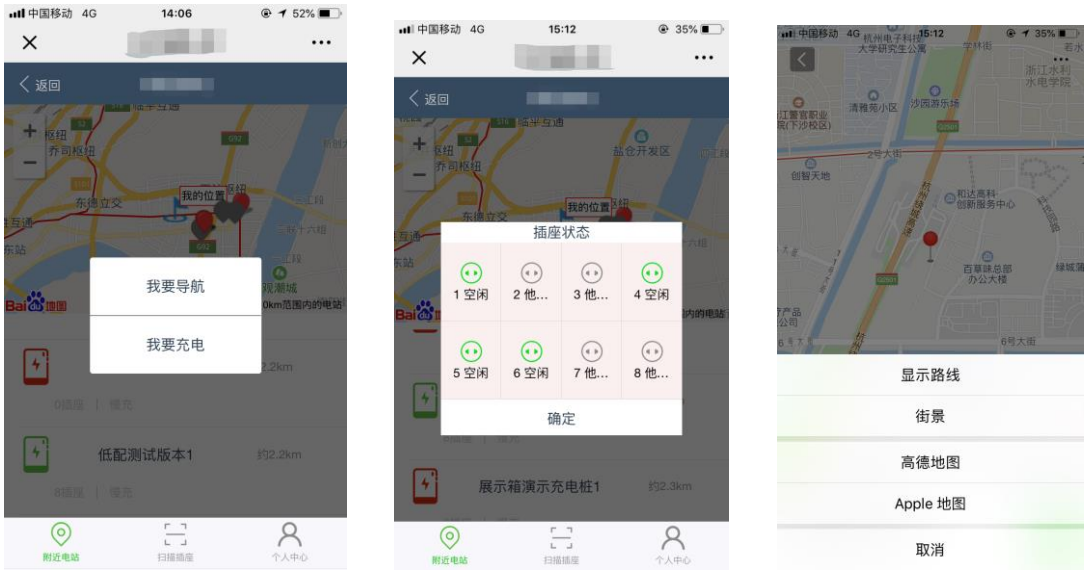
（5）家人模式

家人模式是为不太会使用智能手机的老人考虑的，只需要将插座二维码下方的序列号和对应序号告知子女，子女就可以帮助老人进行远程充电。


图2.30　充电方式-家人模式

## 2.5.4 邻近充电桩搜索及导航

本系统在移动端为用户提供10km范围内的充电桩搜索并设有导航服务，如图2.31所示，在“附近电站”页面，地图标识充电桩位置及在线状态，下方列表显示详细信息，点击列表中对应的充电桩，会出现弹框，用户可选择充电或者导航。点击“我要充电”，用户选择相应插座后，跳转到充电页面；点击“我要导航”，会出现导航选项，用户根据需求选择即可。




（a）附近电站页面　　（b）点击“我要充电”　　（c）点击“我要导航”

图2.31　邻近充电桩搜索及导航

### 2.5.5 应用展示

本文设计的电动自行车智能充电系统已经与杭州 XX 电器有限公司达成合作并进行生产，产品如图 2.32 所示。


图2.32　充电桩产品图

实际车棚安装情况如图 2.33 所示。


图2.33　充电桩实际安装场景

## 2.6 本章小结

本章首先分析了系统的设计需求，然后介绍了电动自行车智能充电系统的架构和组成模

块，并详解说明各部分的设计方案、原理等，最后展示系统实现及应用。

本文设计的智能充电桩系统有如下优势：

（1）安全稳定的充电实现，硬件系统自主识别充电功率，提供多路控制、过流过载保护。

（2）完善的软件服务。Web 管理平台为企业提供可视化服务，管理员可快速导出数据，营收状况一目了然。移动运维端支持随时随地设备录入，提供移动管理，收益情况查询。移动用户端实时显示充电数据，充电完成及充电异常实时微信通知。

（3）可靠稳定的高性能平台支撑。负载均衡及服务器集群实现高可用。设计的改进 JDBC 缓存技术缓解了高并发下的访问压力。

（4）基于微信平台，实现轻量级的应用；依托微信支付，保证资金和交易安全。

本系统充分考虑了不同角色、不同用户的需求，提供多样化服务，系统稳定完善，具有较大的应用价值。

# 第三章 基于四叉树网格划分的邻近充电桩快速搜索算法

在政府的大力推行下，电动自行车充电桩的数量会逐步增加。用户查找附近可达范围的充电桩时，后端程序根据前端传入的用户坐标及数据库中的已有充电桩位置信息，采用常规的计算两点间距离的方式进行筛选，在数据量大的情况下会发生响应延迟，影响系统性能和用户体验。因此，必须寻找一种能快速查询邻近充电桩的方式，将邻近充电桩信息高效准确地呈现给用户。

不管是当前用户位置，还是充电桩位置，都是地理信息的一部分，可以借助地理信息系统（Geographic Information System，GIS）中的空间查询实现以用户为中心，固定距离内的充电桩搜索。

GIS 是一种空间信息系统，它是在计算机软、硬件系统支持下，对整个或部分地球表层空间中的有关地理分布数据进行采集、储存、管理、运算、分析和描述的技术系统[40]。空间查询通过空间索引方法，从数据库中找出符合条件的空间数据，是 GIS 最基本和最常用的功能。

空间数据是对现实世界中存在的具有定位意义的事物的描述[41]，在二维空间下，空间数据即为物体的坐标和范围。空间索引可以排除大量无关的数据，从而提高对空间数据的查询效率。

目前有很多成熟的空间索引技术，大致可分为基于网格的索引、基于树的索引和基于空间填充曲线的索引三类。网格索引将空间划分为均匀的网格，用每个网格对应存储区域，记录网格内的空间对象[42]。采用哈希结构，实现索引和对象之间的映射。网格索引适合用来处理点数据，当空间数据为线、面时，一个空间数据会被多个索引保存，存在冗余。树形结构索引主要有四叉树、R 树、KD-Tree。四叉树索引通过对地理空间递归四分来构建索引，四叉树的每个叶子节点存储了本区域所关联的图元标识列表和本区域空间信息的范围，非叶子节点仅存储本区域空间的范围[43]。四叉树在空间对象分布不均匀时会因为严重不平衡导致查询效率急剧下降。R 树用最小边界矩形包裹节点，解决了数据冗余问题。KD-Tree（k-dimensional 树）是一个 K 维空间中的平衡二叉树，它用 k-1 维的超平面将节点所表示的 k 维空间分成两个部分，查找时，只需在每个内部节点上决定沿哪个走向，直至搜索到叶节点为止。基于填充曲线的索引通过用一条无限长的可以填满整个空间的线实现多维数据对象到一维空间的映射。常见的方法有 Hilbert 曲线、Z 阶曲线。

相较于传统的数据库查询，空间索引大大提升了多维空间数据搜索效率，目前出现的诸多空间索引机制，都存在一些缺陷，如我们的充电桩在地理上分布不均匀，网格索引全局平分的方式会造成很多存储空间浪费。仅仅使用某一种机制无法达到较优的性能。

前面我们介绍过四叉树索引，其原理是不断对空间进行四分，形成一个多层次的树，而

树的节点表示一个空间区域。本文借鉴了四叉树的思想并进行改进，设计了一种查找机制。首先设计 B-GeoHash 编码，对充电桩经纬度二维数据进行降维处理；然后根据空间近似形体与网格的空间关系，对网格区域基于四叉树原理进行自适应非均匀划分，找到满足条件的网格后，用网格编码做前缀匹配查询，得到粗筛的充电桩数据集合；最后，用球面距离对粗筛的数据进行过滤，得到最终精确的结果。

## 3.1 B-GeoHash 编码设计

假设一个与地轴方向一致的圆柱切割于地球，将经纬网投影到圆柱面上，圆柱面展开后，得到一个平面，此时经纬线都是平行直线，且相交成直角。这个投影叫做墨卡托投影，由荷兰地图学家墨卡托(G.Mercator)于 1569 年提出[44]。

根据墨卡托投影得到的平面，我们可以将地球表面划分成一个个网格，用来表示某一区域，GeoHash 就是一种具有网格划分思想的编码，它是一种地理编码，由 Gustavo Niemeyer 发明[45]，其基本思想是将经度和纬度分别按照二分法不断逼近编码，再整合加密，最后得到一维编码表示地球上的二维坐标。对经度和纬度不断二分，只要保证经纬度的划分次数一致，对应到二维空间平面上就是对空间的不断四分，与四叉树的原理相似，所以完全可以使用 GeoHash 编码对四叉树的叶子结点进行编码，来表示空间对象。

以一个坐标 P（119.918555， 29.997688）为例，进行 GeoHash 编码，过程如下。

1、对经度和纬度不断二分编码，小于中间值标记 0，大于中间值标记 1。

表3.1　GeoHash 经度编码

| min | mid | max | bit |
|---|---|---|---|
| -180 | 0 | 180 | 1 |
| 0 | 90 | 180 | 1 |
| 90 | 135 | 180 | 0 |
| 90 | 112.5 | 135 | 1 |
| 112.5 | 123.75 | 135 | 0 |
| 112.5 | 118.125 | 123.75 | 1 |
| 118.125 | 120.9375 | 123.75 | 0 |
| 118.125 | 119.53125 | 120.9375 | 1 |
| 119.53125 | 120.234375 | 120.9375 | 0 |
| 119.53125 | 119.8828125 | 120.234375 | 1 |

表3.2　GeoHash 纬度编码

| min | mid | max | bit |
|---|---|---|---|

| | | | |
|---|---|---|---|
| -90 | 0 | 90 | 1 |
| 0 | 45 | 90 | 0 |
| 0 | 22.5 | 45 | 1 |
| 22.5 | 33.75 | 45 | 0 |
| 22.5 | 28.125 | 33.75 | 1 |
| 28.125 | 30.9375 | 33.75 | 0 |
| 28.125 | 29.53125 | 30.9375 | 1 |
| 29.53125 | 30.234375 | 30.9375 | 0 |
| 29.53125 | 29.8828125 | 30.234375 | 1 |
| 29.8828125 | 30.05859375 | 30.234375 | 0 |

得到经度二进制编码 1101010101，纬度二进制编码 1010101010.

2、将经度编码放在偶数位，纬度编码放在奇数位，得到新的二进制码 α：11100 11001 10011 00110，五个为一组，转换为十进制，不足五位前面补 0，对应得到 28、25、19、6。根据 Base 32 编码表，转换为 GeoHash 码 wtm6。Base32 编码避免了直接使用二进制导致索引过长的问题。

表3.3 base32 编码映射表

| Decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Base 32 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Decimal | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Base 32 | 8 | 9 | b | c | d | e | f | g |
| Decimal | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| Base 32 | h | j | k | m | n | p | q | r |
| Decimal | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Base 32 | s | t | u | v | w | x | y | z |

由编码原理可知，GeoHash 码代表的是一个空间区域，而非一个具体的点，它具有如下特点：

（1）全球唯一性。通过 Geohash 划分出来的区域有且只有全球唯一的空间区域与之对应。

（2）递归性。因为编码过程是不断对区域进行二分，同一区域内同一层级的编码前几位必然是相同的，编码越长，表示的范围越精确；编码越短，表示的范围越大，包含更多长编码区域。如编码 wxs4 区域包含 wxs4j 区域，wxs4j 比 wxs4 表示更精确的范围。我们查找区域内的空间对象时，直接通过前缀匹配进行目标搜索，这种方式在海量数据前表现优异。

（3）编码的一维性。二维经纬度数据用一维字符串来表示，降维处理可以提升检索效率。

目前充电桩业务只在中国大陆范围内，GeoHash 是面向全球的地理编码，会从全球范围

开始编码，这样将造成编码位的浪费和编码精度的降低。本文基于 GeoHash 思想，提出了一种新的编码方案，暂将其命名为 B-GeoHash（Based GeoHash）。

B-GeoHash 码由两部分组成，前 4 位表示空间点的基础经纬度，后几位编码基于 GeoHash 思想，表示更精确的范围。B-GeoHash 编码流程如图 3.1 所示。

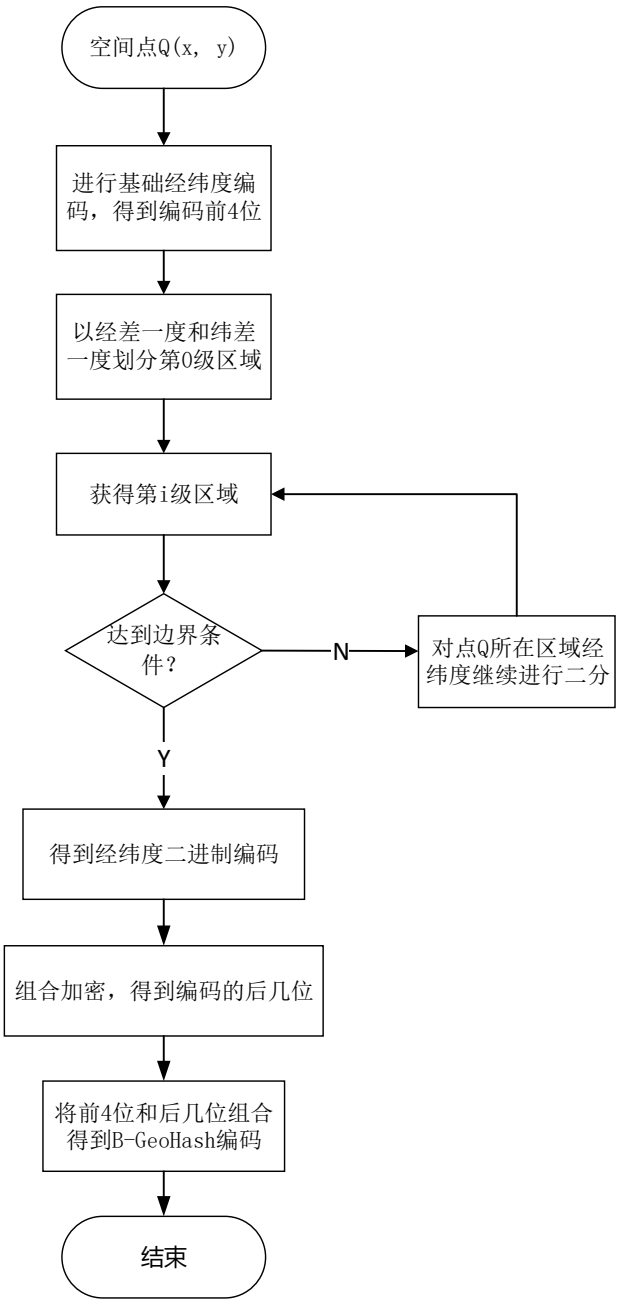



图3.1　B-GeoHash 编码流程图

具体步骤为：

步骤一 对空间点 Q(x, y)（x 为经度，y 为纬度）的经纬度值向下取整，得到点(⌊x⌋, ⌊y⌋)，然后以(⌊x⌋, ⌊y⌋)、(⌊x⌋+1, ⌊y⌋)、(⌊x⌋+1, ⌊y⌋+1) 、(⌊x⌋, ⌊y⌋+1)四个点建立一个经纬度差分别为 1 度的矩形区域，该区域为空间划分的第 0 级区域，如图 3.2 所示。将⌊x⌋、 ⌊y⌋转换为二进制，五个为一组，保证经纬度码的长度相同，不足的在前面补 0，然后按照经度在前纬度在后组

合，进行 Base32 加密，得到 B-GeoHash 码的前 4 位。

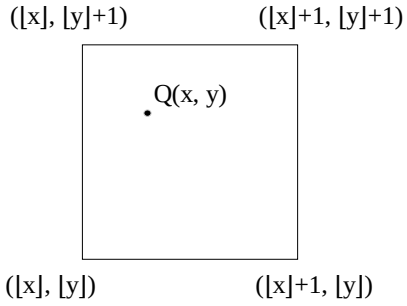



图3.2　构建第 0 级区域

步骤二 在点 Q(x, y)所在区域内，对经纬度同时进行二分，得到点 Q(x, y)所在的第 i 级区域。

步骤三 判断是否达到根据网格划分的边界条件（如划分到十次），如果没有跳转到步骤二，继续二分；否则停止划分，得到点 Q(x, y)所在区域的经纬度二进制编码，按照经度编码在偶数位纬度编码在奇数位组合，然后加密得到 B-GeoHash 码的后几位，具体长度根据边界条件确定。

这样，B-GeoHash 码先由前 4 位定位到某一基础区域，再由后几位具体到该区域内的某一个精确范围。

经纬度每相差一度对应地理上约 111Km 的距离，那么 4 位编码长度表示分别对经纬度进行了十次二分，可以精确到 100m 的范围，几乎可以满足所有充电桩的精度需求。本文选择 10 次作为划分的边界条件，对充电桩进行编码。

仍然以空间点 P（119.918555， 29.997688）为例，演示 B-GeoHash 编码的过程。

1、对点 P 的经纬度向下取整得到基础经度 119 和基础纬度 29，然后以经差 1 度和纬差 1 度建立基础区域，如图 3.3 所示，对基础经度 119 编码得到 00011 10111，对基础纬度 29 编码得到 00000 11101，经度在偶数位纬度在奇数位组合得 00011 10111 00000 11101， 对应十进制 3、23、0、29，Base32 加密转换为 3r0x。

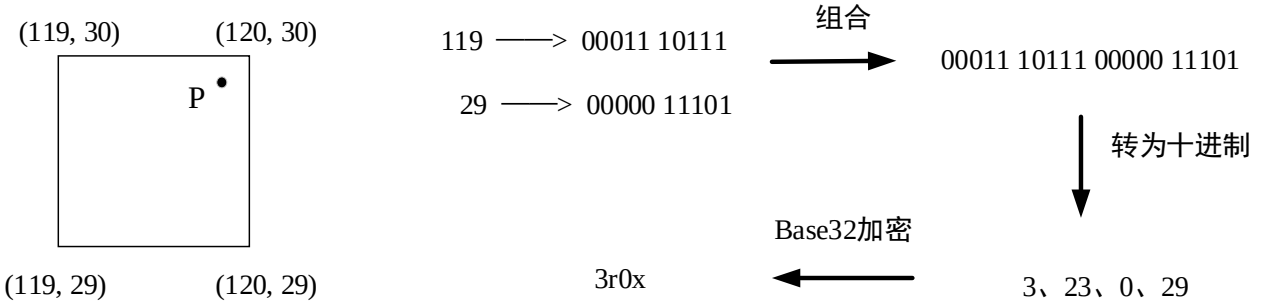



图3.3　基础经纬度编码

2、在基础区域内对点 P 进行二分编码。

表3.4　B-GeoHash 经度编码

| min | mid | max | bit |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 119 | 119.5 | 120 | 1 |
| 119.5 | 119.75 | 120 | 1 |
| 119.75 | 119.875 | 120 | 1 |
| 119.875 | 119.9375 | 120 | 0 |
| 119.875 | 119.90625 | 119.9375 | 1 |
| 119.90625 | 119.921875 | 119.9375 | 0 |
| 119.90625 | 119.9140625 | 119.921875 | 1 |
| 119.9140625 | 119.91796875 | 119.921875 | 1 |
| 119.91796875 | 119.919921875 | 119.921875 | 0 |
| 119.91796875 | 119.9189453125 | 119.919921875 | 0 |

表3.5　B-GeoHash 纬度编码

| min | mid | max | bit |
|---|---|---|---|
| 29 | 29.5 | 30 | 1 |
| 29.5 | 29.75 | 30 | 1 |
| 29.75 | 29.875 | 30 | 1 |
| 29.875 | 29.9375 | 30 | 1 |
| 29.9375 | 29.96875 | 30 | 1 |
| 29.96875 | 29.984375 | 30 | 1 |
| 29.984375 | 29.9921875 | 30 | 1 |
| 29.9921875 | 29.99609375 | 30 | 1 |
| 29.99609375 | 29.998046875 | 30 | 0 |
| 29.99609375 | 29.9970703125 | 29.998046875 | 1 |

当点 P（119.918555， 29.997688）在基础区域内进行十次划分后，满足边界条件，经纬编码 11101 01100，纬度编码 11111 11101，经度编码在偶数位，纬度编码在奇数位，得到新的二进制码 11111 10111 01111 10001，对应十进制 31、23、15、17，Base32 加密为 zrgj。

3、将步骤 1 和步骤 2 中的编码组合得到最终的 B-GeoHash 编码 3r0xzrgj。

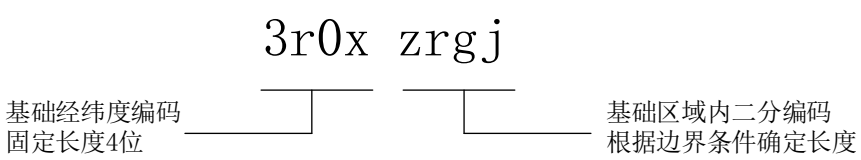



图3.4　B-GeoHash 编码解析

与 GeoHash 编码相比，B-GeoHash 编码完全继承了 GeoHash 编码的优点，实现了：（1）唯一性，B-GeoHash 对应空间的唯一区域；（2）递归性，首先同一区域内，前 4 位的基础区域编码是相同的，后几位编码和 GeoHash 编码一样具有前缀包含的特性，整体的编码符合递

归特性；（3）一维性，实现了一维数据到二维空间的映射。

同时，同样的划分过程，B-GeoHash 编码可以表示比 GeoHash 更精确的范围。同样是对空间点 P（119.918555， 29.997688）进行二分编码，划分十次，GeoHash 编码得到经纬(119.8828125, 120.234375)纬度(29.53125, 30.234375)范围的区域，B-GeoHash 编码得到经纬(119.91796875, 119.9189453125)纬度(29.9970703125, 29.998046875)范围的区域，对于区域内的点，使用 B-GeoHash 编码可以更快地到达目标区域，减少计算次数。

## 3.2 基于四叉树网格划分的快速搜索方法

我们需要检索以用户位置为中心，半径为 R 的圆形区域。B-GeoHash 编码前缀包含的特性就可以实现邻近点搜索，根据用户当前位置 P，结合查询半径 R 确定一个编码等级边界条件，得到 P 在该等级下对应的 B-GeoHash 编码 β，然后用 β 进行前缀匹配，找到该区域内的点，但是这在边界点查找上存在很大问题，如图 3.5，当 P 位于分割线的附近，我们只用区域 A3 的编码去进行匹配，会匹配到大量无关的数据，并且丢失 A1、A2、A4 内符合条件的其他数据。

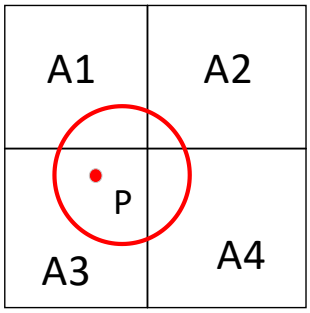



图3.5　B-GeoHash 编码查找边界问题

因此本文不直接采用 B-GeoHash 编码去做搜索，另外设计了一种查找机制，根据四叉树原理划分多级空间网格，利用网格和目标区域的空间关系查找。四叉树划分网格过程与 B-GeoHash 编码过程是基于相同原理的，用 B-GeoHash 编码的思想对划分的网格区域进行编码，就可以利用 B-GeoHash 编码前缀匹配的特性，直接找出该区域内的充电桩。查找过程分为剪枝和过滤两个阶段。

剪枝阶段：目标区域是圆形区域，而我们的网格划分表示的是一个方形的区域。在空间查询中，有一个近似形体的概念，在直接查找空间目标不方便的时候，先用一个简单形体去包围目标，如矩形、圆等。查找的时候先查找该近似形体，再在其内部筛选满足条件的数据。本文中，我们对目标圆形区域建立最小边界矩形 MBR（Minimum Bounding Rectangle），这个空间近似形体为边长 2R 的外切正方形，查找时我们先搜索与 MBR 相交或包含于 MBR 的网格，去掉大部分无关数据。

过滤阶段：计算用户当前位置与空间对象之间的距离，对剪枝阶段得到的粗筛数据进行精准过滤。

### 3.2.1 相关参数和概念

1、参数

表3.6 搜素算法参数表

| 参数 | 定义 |
| --- | --- |
| $R_e$ | 地球半径，单位 km，取赤道半径 6378.137 |
| R | 需要搜索的区域半径 |
| U(x, y) | 用户当前位置坐标，x 为经度，y 为纬度，8 位精度 |
| $C(x_i, y_i)$ | 第 i 个充电桩位置，$x_i$ 为经度，$y_i$ 为纬度，8 位精度 |
| g | 网格编码 |
| $g_{ij}$ | 第 i 级第 j 个网格的编码 |
| $S_{ij}$ | 划分等级为 i 的第 j 个网格，<br>$\sum_{j=0} S_{ij}$为第 i 级网格合集 |

2、空间关系

查询区域与网格之间存在相交、包含、被包含三种空间关系。（1）被包含关系，查询区域被当前网格包含。（2）相交关系，查询区域与当前网格相交。（3）包含关系，查询区域包含当前网格。如图 3.6 所示，红色矩形表示查询区域。

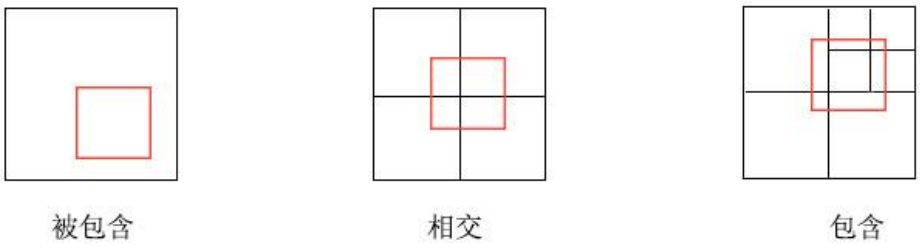



图3.6 查询区域与网格空间关系

3、边界条件设定

划分次数越多，网格越小，数据更精确。但是节点数目和树的高度都会增加，导致剪枝时间增加。网格分裂越多，剪枝代价越高，过滤求精代价越低。需要在剪枝和过滤之间寻找一个平衡。

MBR 的大小是影响网格划分层级的一个重要因素，空间经纬度相差 1 度，地理距离为 111km，那么至少需要划分$[\log_2 111/2R]$级网络，这一级网络保证的是一个接近 MBR 大小的区域，我们在此基础上，再增加三次划分，基本可以满足要求。

另外，我们在对充电桩进行 B-GeoHash 编码时，只进行了 10 级划分，如果查找的时候网格划分大于 10 级，得出的网格编码长度比充电桩编码长度长，是匹配不到充电桩数据的，所以在划分十次后，无需继续划分。

综上，得出网格划分层级 i 的边界条件为：

$$i < \log_2 \frac{111}{R} + 2$$
$$i \leq 10 \qquad (3.1)$$

### 3.2.2 剪枝阶段

采用网格叠加法，将网格与目标区域叠加，从第 0 级网格开始依次进行相交分析，记录子网格与目标范围的空间关系，进行相应处理，直至遍历完所有网格。

本算法通过队列 GridQueue 实现网格的遍历和层级划分，建立空间实体与多级网格的对应关系，并将符合条件的充电桩放入 stationList 中。

算法具体流程如下：

步骤一 根据用户当前位置 U(x, y)和搜索半径 R，绘制边长 2R 的最小边界矩形 MBR（Minimum Bounding Rectangle）。

步骤二 对 U(x, y)的经纬度向下取整⌊x⌋、 ⌊y⌋，得到基础网格，即第 0 级网格 $S_{00}$。将 $S_{00}$ 放入队列 GridQueue。同时考虑到用户位于基础经纬度边界的情况，将周围八个同级基础网格放入队列，如图 3.7 所示。

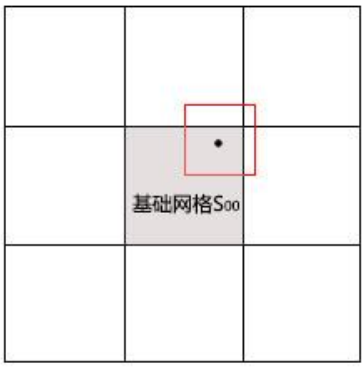



图3.7 用户位于基础网格边界情况

步骤三 取出队头元素 $S_{ij}$，判断 MBR 与 $S_{ij}$ 的空间关系。如果 MBR 与 $S_{ij}$ 相交或者被包含于 $S_{ij}$，判断边界条件，若$i < \log_2 \frac{111}{R} + 2$且$i < 10$，将网格四分，得到四个子网格，将这些网格放入 GridQueue 队尾；否则停止划分，进入步骤四。如果 MBR 包含一个或多个网格，进行步骤四。如果 MBR 与 $S_{ij}$ 相互独立，不存在空间联系，直接从队头删除 $S_{ij}$。

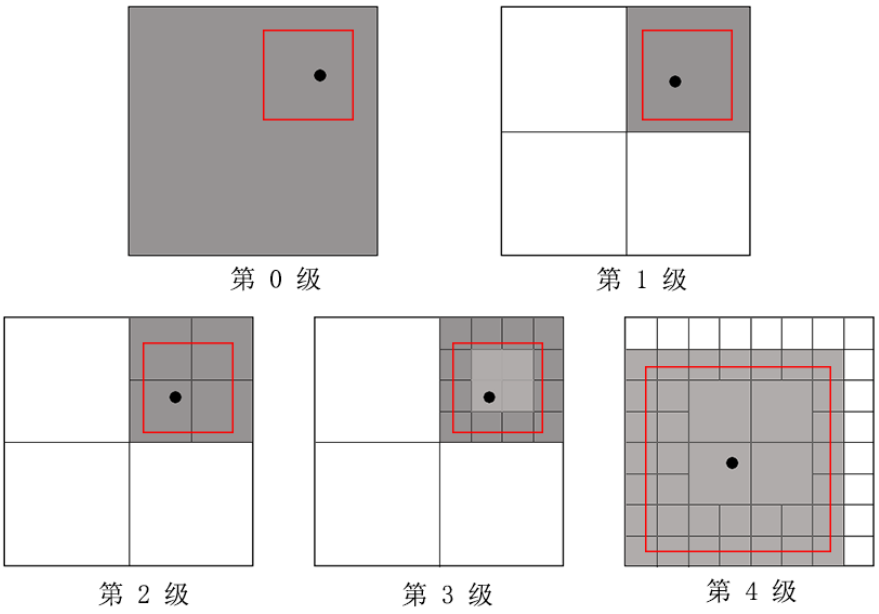



图3.8　多级网格查找过程

步骤四　根据网格编码在数据库中进行前缀匹配查询，得到这些区域内的充电桩数据，放入 stationList 中。

步骤五　依次处理 GridQueue 中的数据，直至 GridQueue 为空，结束搜索。

### 3.2.3 过滤阶段

剪枝阶段得到的 stationList 中，存在一些不满足条件的数据。如图 3.9，这些误差数据主要来自两个方面：（1）剪枝过程涵盖的区域与 MBR 存在误差。（2）我们实际要查找的是一个圆形区域，MBR 与目标区域有误差。所以，确定了可能满足条件的空间对象集合 stationList 后，要根据两点间距离进行精准过滤。

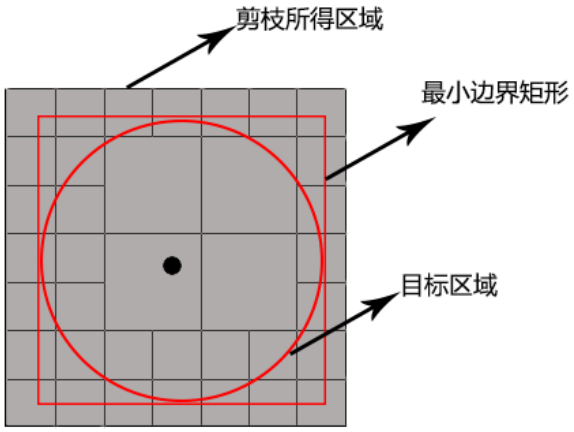



图3.9　剪枝误差来源示意图

地球表面是一个有弧度的平面，在进行数据过滤时，我们计算充电桩 $C(x_i, y_i)$与用户当前位置 U(x, y)两点间的距离，采用式（3.2）表示的地球球面距离公式，而不是计算直线距离。

$$\widehat{UC_i} = R_e * \arccos\left[\cos y \cos y_i \cos(x - x_i) + \sin y \sin y_i\right] \quad (3.2)$$

根据剪枝和过滤阶段分析，总结查找算法的整体流程如下：

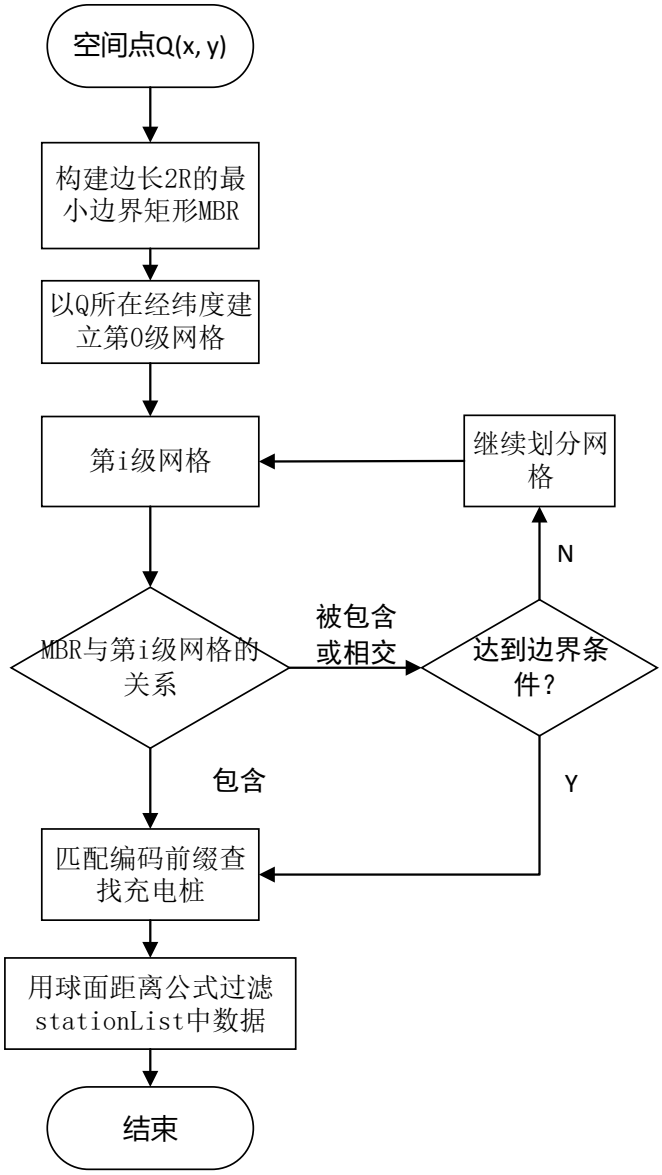



图3.10 查找算法流程

## 3.3 算法分析

充电桩 B-GeoHash 编码属于数据预处理部分，不影响查询的时间复杂度。本文只讨论剪枝、过滤两个阶段的时间复杂度。

搜索过程中，用空间叠加法对 MBR 和网格的空间关系进行逐层查找，设划分过程中第 i 级网格的数量为 $M_i$，对每一层的网格需要遍历判断空间关系并处理，总的次数为

$$\sum_{i=0}^{10} M_i \quad (3.3)$$

剪枝阶段的时间复杂度为 O(M)。

过滤阶段，设 stationList 中的数据量为 K，每次只需要进行一次时间复杂度 O(1)的计算，过滤阶段的总时间复杂度为 O(K)。

本文提出的算法总时间复杂度为 O(M+K)。设数据库中充电桩数量为 N，采用传统的数据库查询方式，计算每个充电桩和用户位置的距离进行搜索，其时间复杂度是 O(N)。本文的算法，网格划分最多 10 次，过程中产生的 M+K 是远远小于庞大的充电桩数据量 N 的，相比之下具有更高的查询效率。

本文提出的算法具有如下特点

（1）基于 GeoHash 编码原理，设计适应当前应用场景的 B-GeoHash 编码，同一区域内的充电桩编码前缀相同，搜索时通过前缀匹配可实现同一区域内充电桩的快速查询。

（2）用四叉树划分空间多级非均匀网格，减少了数据冗余。网格划分过程与 B-GeoHash 编码保持一致，保证了查找与编码的一致性。

## 3.4 测试分析

本章设计了一种邻近充电桩搜索算法，为了验证该方法的可行性与性能优势，进行了两个方面的测试：一是准确性测试，以常规的计算用户与充电桩之间距离筛选的方法为标准，测试本章提出的搜索方法的搜索准确性；二是耗时测试，对比相同场景下常规方法与本章提出的算法得到结果所用时间，分析算法性能。

目前电动自行车的续航能力集中在 40-60km 范围，最大一般不超过 80km。用户需要充电时，电动车可行驶的剩余里程不多，并且距离过远用户前往的意愿较低，综合考量暂定搜索 10km 范围内的充电桩。

实验时采用阿里云的云服务器和 MySQL 数据库，服务器为 2 核 8G 的 64 位 Linux 系统，MySQL 配置为 2 核 4G。

### 3.4.1 准确性测试

#### 3.4.1.1 测试方案

在全国范围内随机生成 10 万个充电桩数据，对充电桩的位置信息进行 B-GeoHash 编码预处理；随机生成 100 个用户位置坐标并按照 1-100 进行编号；将通过计算用户与充电桩之间距离筛选的方法作为常规方法，记录常规方法与本文方法查询这 100 个用户附近 10km 范围内充电桩的结果。

#### 3.4.1.2 测试数据

以用户编号为横坐标，查询结果数为纵坐标，生成图表，如图 3.11 所示，蓝色数据代表本文方法搜索结果，红色数据代表常规方法搜索结果，绿色数据表示两者的差值。

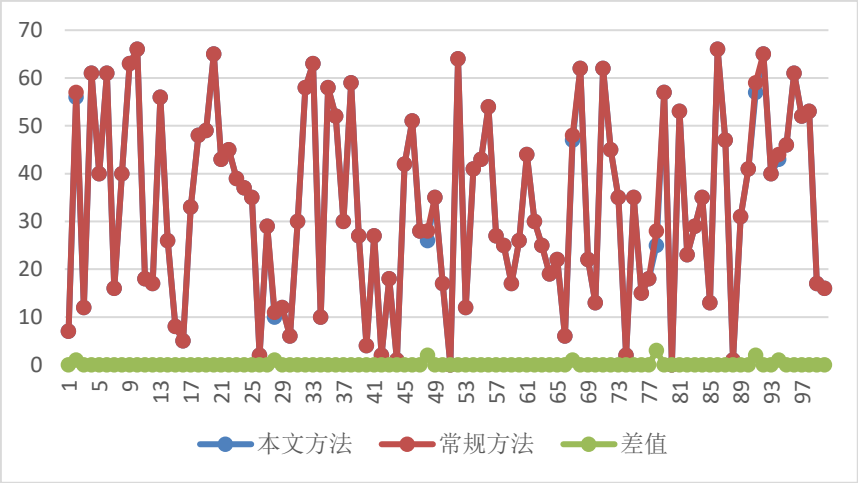



图3.11 搜索结果对比

#### 3.4.1.3 测试分析

图 3.11 中，本文方法和常规方法的数据点大部分都是重合的，只有极个别情况，本文方法比常规方法少一两个数据，这点根据绿色的数据也可以很清晰地看出。这样的情况是完全合理的。本文方式剪枝之后的过滤阶段，也采用计算两点间距离的方式对粗筛数据进行精确过滤，所以本文方式得到的数据一定是包含在常规方式筛选出来的数据集中的，有极个别误差是因为有些充电桩位于网格划分的边界上，剪枝阶段被丢失，造成最后结果的缺失。但是这种情况出现的概率非常低，我们查询得到的 3336 条充电桩数据中，只有 11 条数据缺失，且这些数据点位于边界上，对用户来说距离远，数据价值低，这些数据极小概率的缺失不会对系统造成什么影响，可以认为本文算法搜索得到的数据完全准确。

### 3.4.2 耗时测试

#### 3.4.2.1 测试方案

通过程序模拟，在全国范围内随机生成 1 万、2 万、3 万、5 万、10 万、15 万、20 万、30 万个充电桩数据，进行 B-GeoHash 编码预处理；再随机生成 100 个用户位置坐标；将不同充电桩数据量作为变量，统计常规方法和本文方法查询 100 个用户附近 10km 范围内的充电桩所用的平均时间。

#### 3.4.2.2 测试数据

测试得到表 3.7 中数据，对比如图 3.12 所示。

表3.7　搜索耗时记录表

| 充电桩数量（个） | 常规方法平均耗时（s） | 本文方法平均耗时（s） |
|---|---|---|
| 1 万 | 2.77 | 0.231 |
| 2 万 | 2.89 | 0.363 |
| 3 万 | 2.93 | 0.425 |
| 5 万 | 3.06 | 0.440 |
| 10 万 | 3.34 | 0.564 |
| 15 万 | 3.46 | 0.594 |
| 20 万 | 3.85 | 0.631 |
| 30 万 | 4.66 | 0.665 |

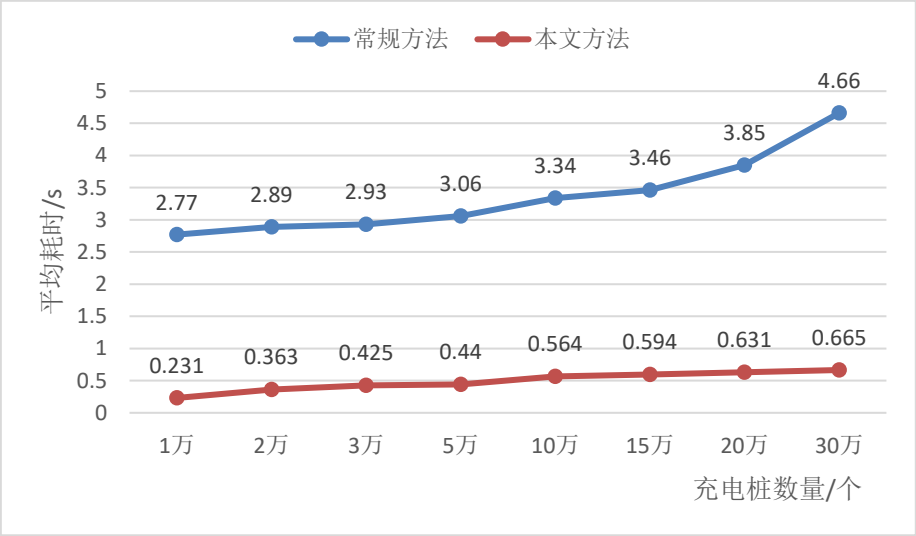



图3.12　搜索耗时对比图

#### 3.4.2.3 测试分析

从图 3.12 可以看出，随着数据量的增加，两条曲线都呈增长趋势。但是本文提出的算法一直保持很低的查询时间，不超过 1 秒，且增长幅度不大；而常规查询耗时 2s 以上，数据越多，耗时增长幅度越大，30 万数据量时 4.66s 才得到结果，严重影响用户体验。本文方法查询时间均不超过 1 秒，按照网页响应 2/5/10 秒的划分标准，在让用户满意的范围内，符合我们的设计需求。

经过计算分析，本文算法搜索所用时间比常规方法减少了 84%，对系统性能有较大改善。

## 3.5 本章小结

本章首先说明了直连经纬度的常规查询方式弊端，引出空间查询技术，然后对空间索引、网格划分、编码设计等进行了充分研究，提出一种高效的邻近充电桩查找算法，该方法在充

电桩地理位置分布不均匀、数据规模大等情况下具有很优的查询效率。

用自主设计的 B-GeoHash 编码进行充电桩编码，对空间数据降维，并用前缀包含的特点实现区域内充电桩的关联。查找时，分为剪枝和过滤两个阶段。剪枝阶段，根据网格与 MBR 的空间关系，对基础区域递归地进行四分构建多层次的网格系统，用空间叠加法找出满足条件的网格，减少了数据冗余，增加了查询命中率。过滤阶段，用球面距离筛选得到精准数据。同时在细节上增加设计，解决了边界点的问题。

实验表明，本文提出的邻近充电桩搜索方法比传统查询方法减少了 84%的时间，具有很高的查询效率。

# 第四章 基于 JDBC 的数据缓存研究及实现

本文设计的电动自行车智能充电系统，存在高并发的情况。如早晨上班时间段，大量用户同时访问，给服务器带来较大压力，降低系统性能。表现在前端层面就是页面的加载慢，用户长时间得不到响应，影响使用体验。在手机网速不稳定的情况下，问题更加严重。提高页面响应速度、减少用户等待时间是系统优化的重点。

页面加载时长由静态资源大小和数据操作两部分决定。目前大部分前端应用都是单页面应用，静态资源大小只影响首屏加载速度，而数据操作则时时刻刻影响着系统响应性能。系统中业务逻辑处理由业务云完成，业务云中与数据处理有关的是业务服务器和数据库。系统采用前后端分离的模式开发，用户进行操作，客户端向业务服务器发送请求，请求往往伴随着对数据库的操作，业务服务器需要与数据库建立连接，数据库执行 SQL 指令后返回结果，再由业务服务器处理并回传给客户端。

业务服务器与数据库进行通信需要额外开销，当用户存在大量重复请求，如果每次都进行一次连接查询回复的操作，会带来大量资源浪费。可以借助服务器缓存技术，将相关数据按照一定的规则存储在内存中，在用户访问时直接从缓存读取[47]，而不重新建立数据库连接，就可以减少磁盘访问次数，缓解系统压力，缩短用户等待时间。

目前已经发展出很多服务器缓存技术，比较成熟的 Redis、Memcache 可以实现复杂的缓存组织和管理[48]，但是开发成本高，并且 Redis、Memcache 都是系统的额外插件，本身就会带来资源消耗。JDBC 缓存技术通过在 JDBC 上扩展功能，更轻便地实现了数据缓存，在中小型项目中具有优势。JDBC 缓存技术将符合规则的 SQL 请求结果集缓存在服务器内存中，下一个请求到达时，先对请求进行过滤，判断缓存中是否有相应结果集，如果有直接返回，避免重复与数据库建立连接。但是现有 JDBC 缓存技术发展还不成熟，仅仅只是简单的实现，没有实际应用，并且对数据置换、一致性维护问题研究不足，导致 JDBC 缓存的整体性能不高。

本文在现有 JDBC 缓存技术基础上，设计了合理的数据组织和存储结构，在智能充电系统中实现了 JDBC 缓存的应用，并着重对内存达到上限时的数据置换策略进行研究，提出基于 LRU（Least Recently Used，最久未被使用）的改进缓存置换算法，提高了缓存命中率。对于数据库更新导致在缓存中读取到脏数据的问题，设计了一致性维护机制保障缓存数据准确性。

## 4.1 JDBC 原理及实现机制

JDBC 即 Java 数据库连接（Java Database Connectivity），是一种 Java API[46]，用来实现应

用程序对不同数据库的统一操作。它是一种基础接口，本身只能实现三种功能：建立数据库连接，发送操作语句和传输响应结果。但开发人员可以在其上扩展高级接口和功能，进行自定义开发，本文方案就是通过对 JDBC 相关接口的修改和扩展实现的。

JDBC 由以下四部分组成：Java 应用程序层，JDBC API 层，JDBC 驱动程序管理器层和数据库层[49]，架构如图 4.1 所示。Java 应用程序调用封装好的 JDBC API 接口与驱动管理器通信，驱动管理器选取相应驱动程序，实现与数据库的通信。

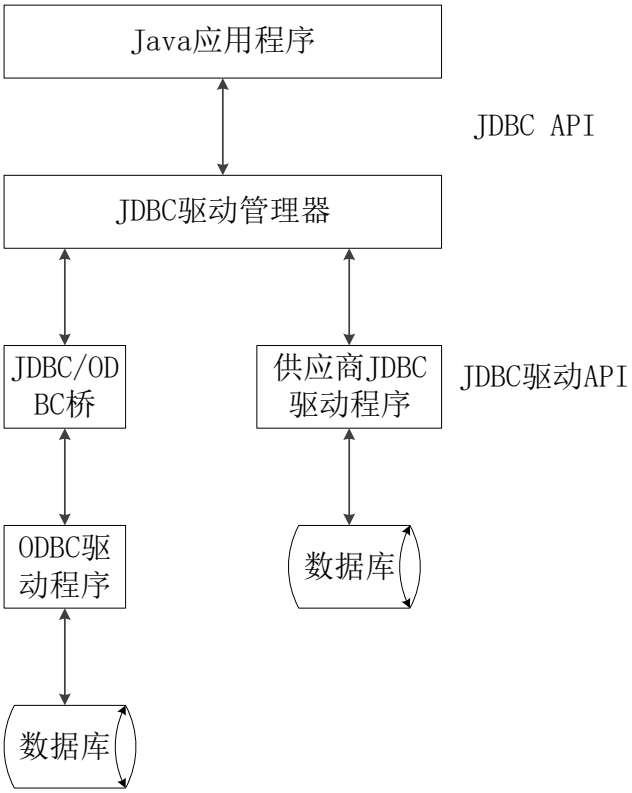



图4.1 JDBC 架构

JDBC API 层是 JDBC 架构的核心，它提供两种访问模型：两层模型 C/S（Client/Server）和三层模型 B/S（Browser/Server）。两层模型中，客户端直接连接数据库，进行数据操作。三层模型中，客户端和数据库之间有一个服务器作为中间层，客户端的请求经由服务器转发给数据库处理。三层模型中的中间层可以方便我们增加自主控制设计，进行功能扩展，本文采用的是 B/S 三层模型。

JDBC API 定义了一系列抽象的 Java 接口[50]，重要的接口有：DriverManager：装载驱动程序，支撑数据库连接的建立；Connection：表示某一数据库连接对象，该对象在上下文中创建其他 JDBC 对象，并管理数据库事务[51]；Statement：用来构造并向数据库发送 SQL 指令；PreparedStatement：Statement 的子类，可以对 SQL 进行预编译；ResultSet：表示 SQL 指令执行得到的结果集，提供遍历数据集合的方法。

JDBC API 核心接口之间的关系如图 4.2 所示。

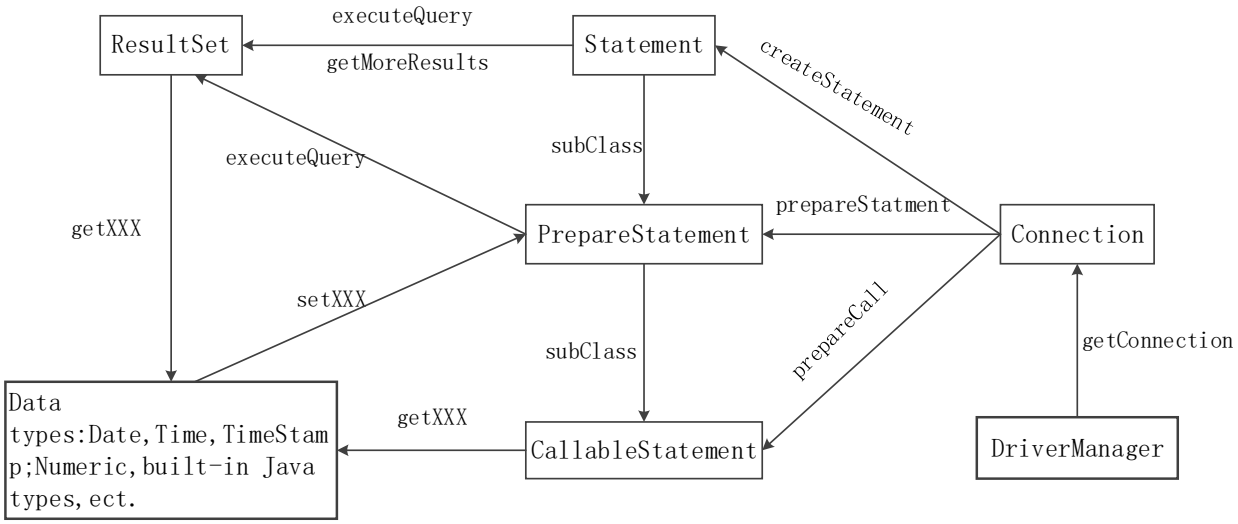



图4.2　JDBC API 核心接口关系

JDBC 通过 API 与数据库进行数据存取的过程包括：装载驱动程序、建立数据库连接、执行数据库操作、处理结果集、关闭连接释放资源，流程如图 4.3 所示。

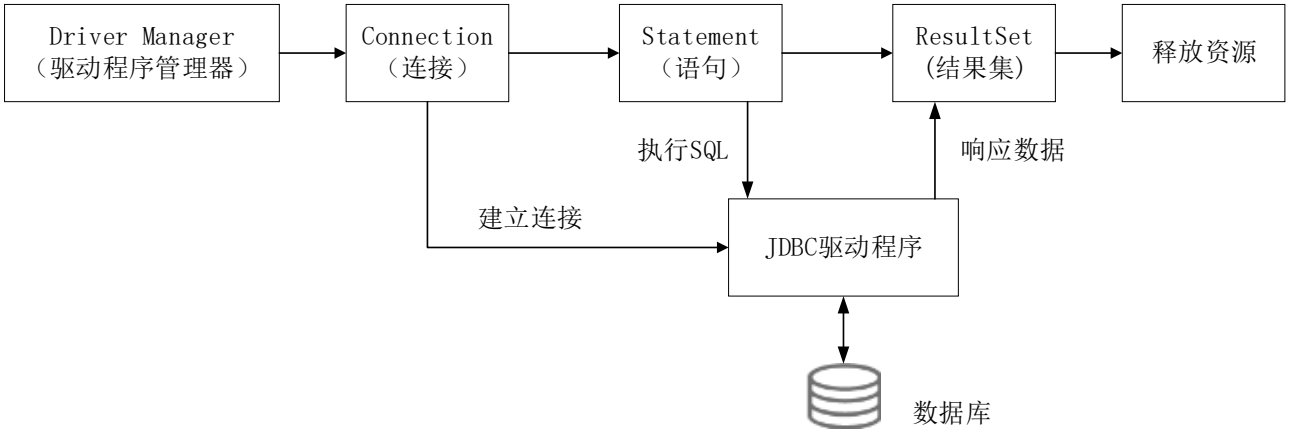



图4.3　JDBC 操作数据库流程

## 4.2 JDBC 缓存设计及实现

根据现有 JDBC 缓存技术的原理，本文设计了合理的存储组织结构、缓存置换机制和一致性维护机制，实现了一个改进的 JDBC 缓存数据管理模型，如图 4.4 所示。

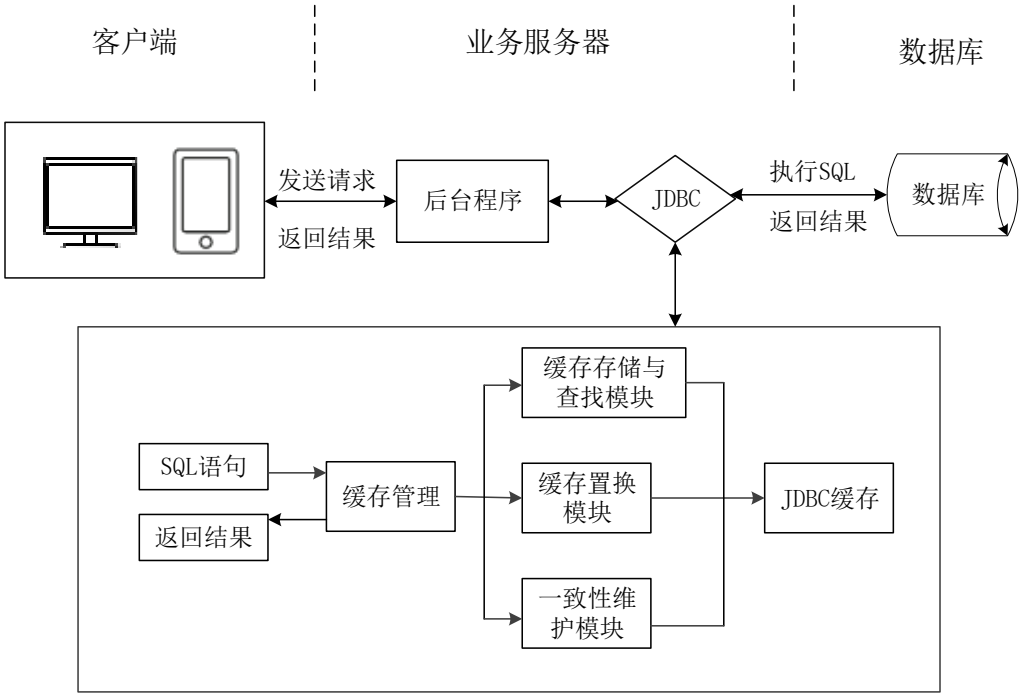



图4.4 JDBC 缓存数据管理模型

缓存管理包含三个功能模块：缓存存储与查找模块、缓存置换模块、一致性维护模块，这三个模块保证了缓存的有序组织、内存的合理分配以及缓存数据的准确性。

缓存存储与查找模块中封装功能接口，实现数据的存储、过滤和管理。缓存置换模块在缓存容量达到阈值时按照改进的 LRU 算法进行缓存置换，实现内存的合理分配。一致性维护模块中设计了一种维护机制，将多次更新操作收集到一起，遇到触发条件再批量执行，提高了缓存维护的效率。

### 4.2.1 缓存对象

数据库操作主要是增删改查四类，增删改操作涉及到数据的变更，使用缓存会导致一致性维护困难，本文的缓存仅考虑对 SQL 查询指令结果集的缓存。如充电页面“功率区间-单价”对照表，每个用户充电时进入充电页面都会请求这一数据来确定当前电池对应的计费区间，将这一数据缓存可以在高峰时段减轻服务器压力。

同时，被缓存的 SQL 查询指令结果集应符合以下命中规则：（1）不含 WHERE 条件的 SELECT 语句，WHERE 条件项复杂多变，难以统一规划。（2）不含 COUNT()、NOW()等特殊函数，这些函数每次计算得到的数据不一致，对于这些数据的缓存是没有价值的。

### 4.2.2 缓存存储与查找

#### 4.2.2.1 存储结构

缓存数据的读取效率对整个缓存性能有重要影响。key-value 结构以“键—值”的形式存储数据，获取数据时，根据 key 直接读取对应的 value，时间复杂度为 O(1)。HashTable 是一种常用的键值对存储数据结构，本文采用 HashTable 存储缓存数据，实现缓存的高效读取。

SQL 语句执行后得到一个结果集 ResultSet，用 SQL 语句作 key 值，把查询结果集及相关属性当作缓存项作为对应的 value 值，存储在 Java 虚拟机堆内存中。存储前使用 Java 中 String 类提供的方法对 SQL 语句做初步的处理，去除空格、统一关键字大小写，避免因为书写习惯造成的操作目的相同但 SQL 语句字符串不同的问题。

缓存项中除了 SQL 语句对应的 ResultSet，还需要保存一些其他属性信息，来配合实现数据置换机制、一致性维护机制，缓存项中的属性如表 4.1 所示。

表4.1　JDBC 缓存项属性表

| 属性 | 定义 |
| --- | --- |
| sql | 缓存项对应的 SQL 语句规范化处理后的字符串 |
| result | SQL 语句执行后得到的结果集 |
| hitTimes | 缓存命中次数，配合缓存置换机制 |
| list | 缓存项对应的更新记录集合，配合一致性维护机制 |

list 属性记录缓存项对应的 SQL 更新操作，即数据的增删改，这一属性用来进行缓存一致性维护。采用 ArrayList 数据结构存储，可以保证存储的有序性，当需要进行一致性维护时，有序地执行 SQL 语句，保证了一致性维护的准确。

HashTable 存储的数据是无序的，本文设计基于 LRU 算法的数据置换机制，需要区分缓存项的访问时间先后。虽然可以通过记录最近访问时间实现，但是当需要找出最久没有访问的缓存项时，必须遍历散列表，时间复杂度高。采用链表结构，将每次命中的缓存项移动到链表的头部或者将新增的缓存项添加到链表头部，这样，整个链表按照缓存项的访问时间先后顺序进行组织，我们想要淘汰最久没有使用的数据时，淘汰掉尾结点即可。

当缓存容量达到阈值，需要进行缓存淘汰时，要进行删除尾结点和插入头结点的操作，如果采用单向链表，移动中间节点到头结点，我们需要知道中间节点前一个节点的信息，单向链表就不得不再次遍历获取信息，而使用双向链表，增加了前驱指针，记录着上一节点信息，无需再次遍历。

综上，本文采用双向链表+HashTable 的结构存储缓存数据，读取效率高，数据置换方便，结构如图 4.5 所示。

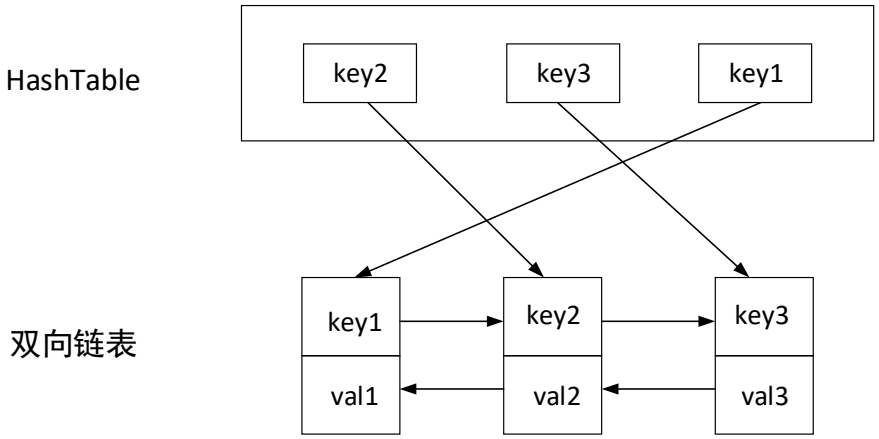



图4.5　JDBC 缓存数据存储结构

#### 4.2.2.2 缓存存储与查找机制

JDBC 缓存将数据库中的数据前置到服务器中，所以需要在 JDBC 连接数据库前进行处理。JDBC 中，Connection 连接对象由 DriverManager 接口的 getConnection 方法创建，本文通过修改扩展 DriverManager 接口实现请求过滤。当新的结果集产生，需要判断是否满足缓存命中规则。Statement 接口用 executeQuery 方法执行数据查询的 SQL 语句，返回查询得到的 ResultSet 结果集对象，因此，将数据缓存判断和存储设计在 Statement 接口中。

首先，扩展 DriverManager 接口，根据传入的 SQL 语句，检查缓存中是否存在对应的缓存项，如果存在，从缓存中取出相应结果集；否则，创建 Connection，连接数据库，在 Statement 接口中添加标志变量 flag，flag 根据 SQL 语句是否满足命中规则置为 true 或 false，然后用 executeQuery 方法执行 SQL 语句得到结果集返回给应用服务器，同时，根据缓存标志变量 flag 判断是否进行缓存。如果 flag 为 true，缓存对应的结果集。整个缓存存储与查找的流程如图 4.6 所示。

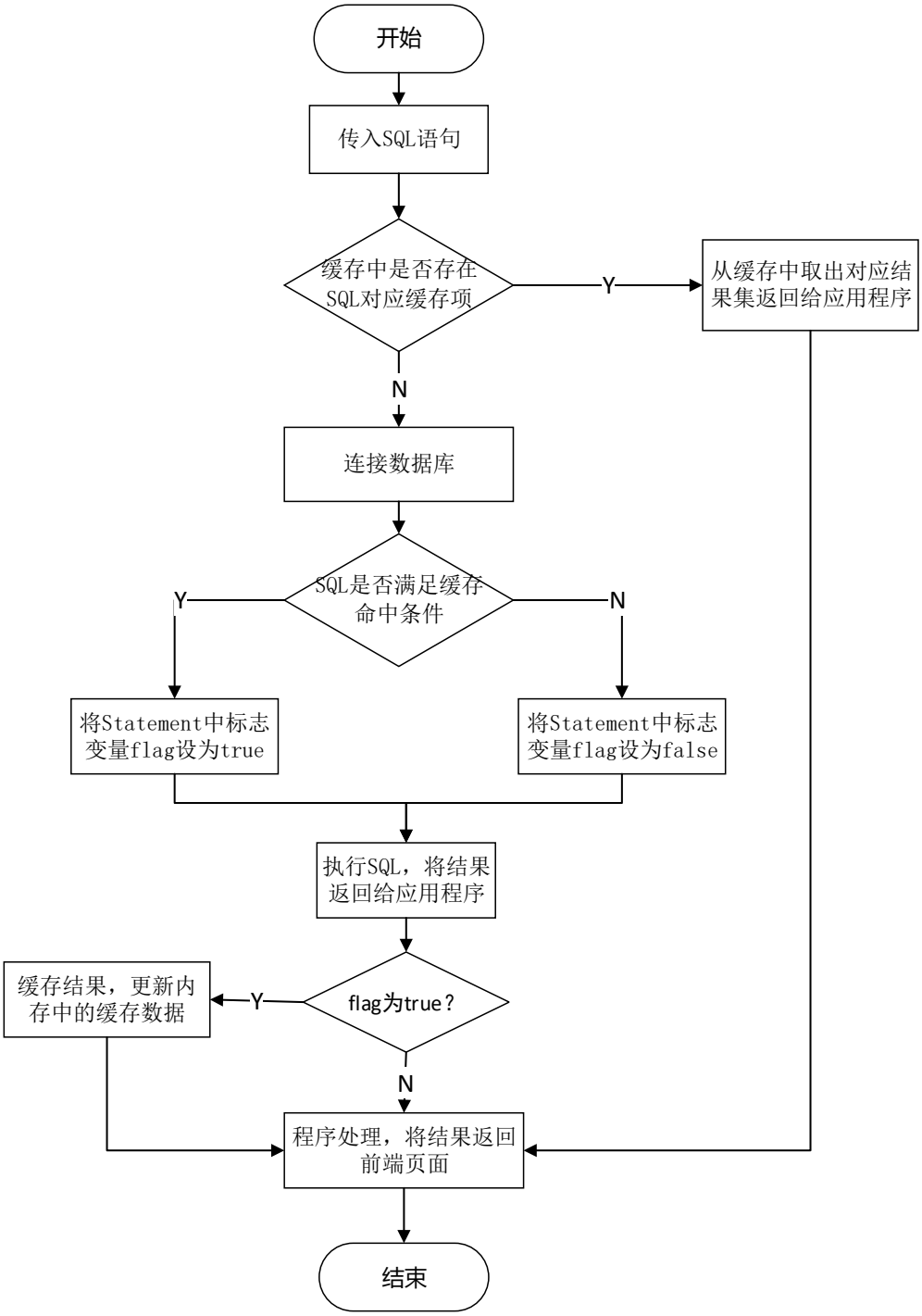



图4.6 JDBC 缓存工作流程

### 4.2.3 缓存置换

内存空间有限，且内存占用过多会影响性能，需要合理控制缓存容量。当需要加入新的缓存项，但缓存容量已达到阈值时，要选择缓存中的一些数据进行替换。合理的缓存置换策略很重要，如果替换出的数据是经常被用到的热点数据，就会造成缓存命中率的下降，影响

整体的缓存性能。在设计时，要尽量让那些使用次数较少、很久没有使用的价值较低的数据被置换出去。

常见的数据置换策略有 FIFO（First Input First Output）、LRU、LFU（Least Frequently Used）。FIFO 按照先入先出的规则，优先清理最先被缓存的数据对象，它对所有缓存数据一视同仁，只按照缓存创建的先后顺序淘汰，对热点数据没有区分度。LFU 优先清理命中次数低的数据，这样使用次数多的缓存项可以长久保存在内存中，缓存命中率比 FIFO 算法高，但是 LFU 没有办法对一些以前经常使用现在几乎不用的缓存项负责，实际中使用较少。LRU 是最先淘汰最久没有使用的数据，实现简单，但是偶发性的操作可能会导致不常使用的数据把热点数据挤出缓存，造成缓存污染。本文改进 LRU 算法设计了数据置换策略，减轻了缓存污染问题，提高了缓存命中率。

LRU 算法通过链表实现，当命中缓存时，将该节点移动到链表头部；当新增缓存时，将该节点置于链表头部。这样，链表上的节点按照访问时间先后顺序排列，需要数据置换时，淘汰尾结点，添加头结点即可。过程如图 4.7。

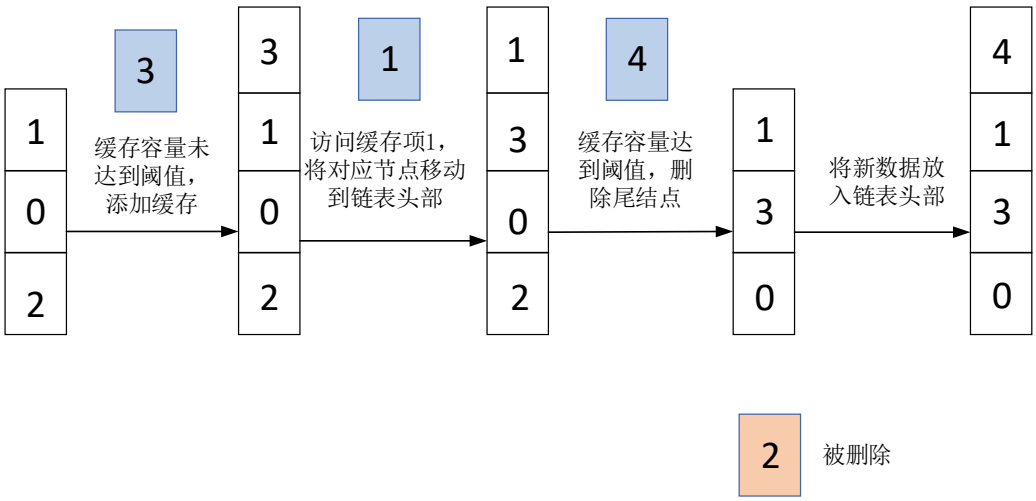



图4.7　LRU 数据置换流程

LRU 算法存在缓存污染问题，如图 4.8，灰色圆形中记录的是缓存项的访问次数，加入缓存项 4 的时候因为内存不足，淘汰了缓存项 2，但是缓存项 2 的命中次数很多，被淘汰之后会降低整个缓存的命中率，不常用的数据把常用数据挤出缓存，就是缓存污染现象。

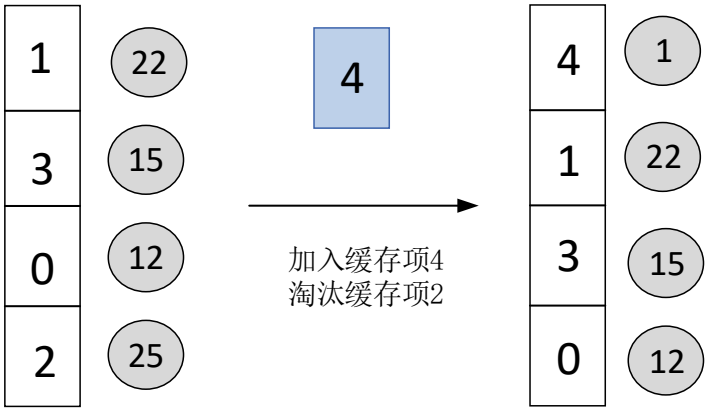



图4.8　LRU 缓存污染问题

为了解决缓存污染问题，本文对 LRU 算法进行了改进，采用两个双向链表进行缓存置换管理，两个链表分别表示热数据区和冷数据区，用链表 H、链表 C 指代。当有新增缓存项时，将该缓存项放入链表 C 的头部，缓存项中的 hitTimes 属性记为 1。链表 C 按照 LRU 规则进行缓存淘汰。每次命中缓存项时将该项的 hitTimes 加 1，当某个缓存项的 hitTimes 达到设定的值 k 时，从链表 C 中删除该项，放入链表 H 的头部。如果命中的缓存项在链表 H 中，按照 LRU 规则进行管理和淘汰，但是淘汰出的缓存项不是从内存中删除了，而是插入到链表 C 的头部。缓存只从链表 C 的尾部删除，k 的值根据具体业务需求确定。总结起来主要包含以下几点：

（1）新缓存放入链表 C 头部。

（2）链表 H 和链表 C 都按照 LRU 规则管理缓存。

（3）当链表 C 中的缓存项访问次数达到一定量 k 时，将数据从链表 C 中删除，加入链表 H 头部。

（4）链表 H 淘汰的数据放入链表 C 的头部。

（5）链表 C 淘汰的数据从缓存中删除。

改进的缓存置换机制根据缓存命中次数，分成冷热数据区存储数据，只从冷数据区淘汰数据，尽量避免置换出热数据，提高了整体的缓存命中率。示意图如图 4.9 所示。

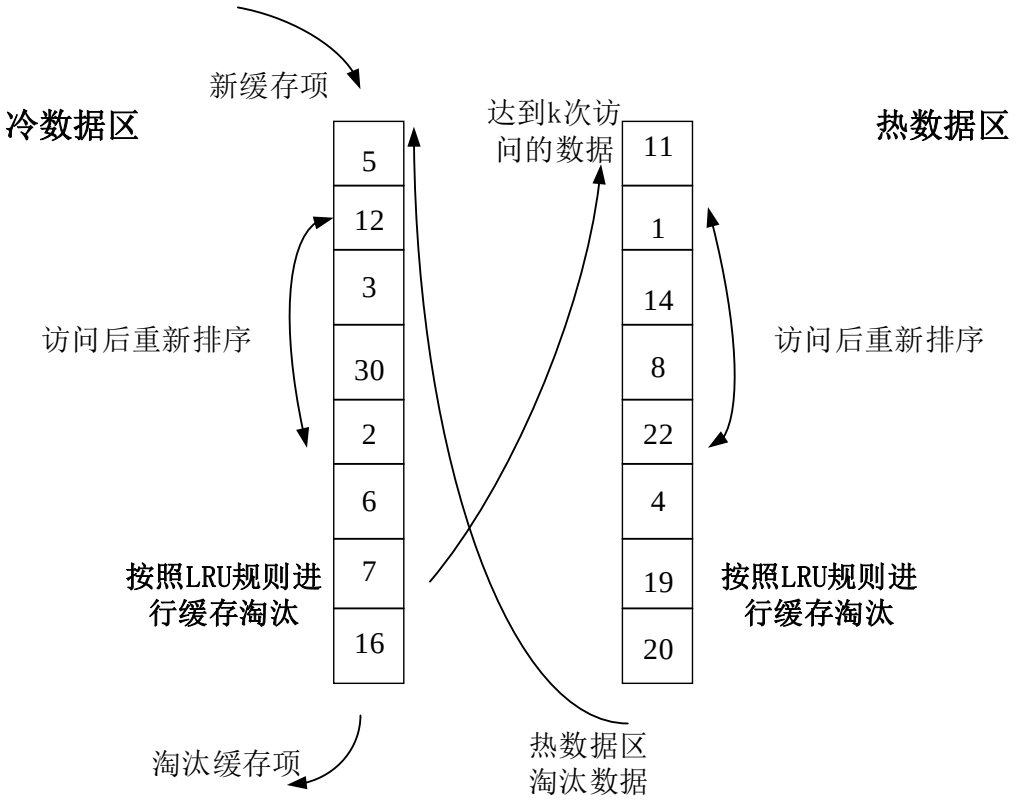



图4.9　缓存置换机制

当有新的缓存项需要加入内存，本文置换机制工作流程如图 4.10 所示，如果缓存容量达到上限，需要进行数据置换，将冷数据区的尾结点淘汰，将新数据放入冷数据区头部。

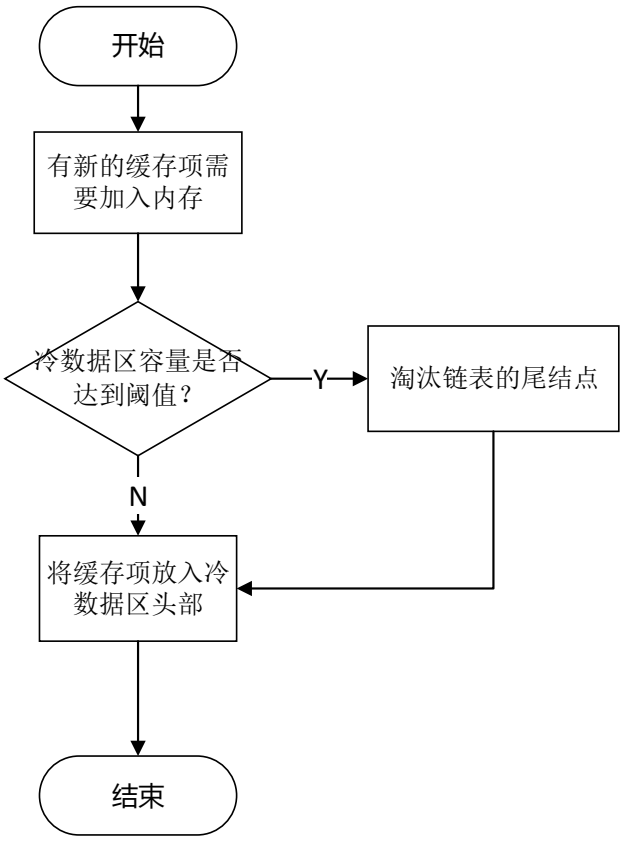



图4.10 缓存置换机制工作流程

当命中缓存项时，虽然不需要进行缓存淘汰，但置换机制需要对冷热数据区的数据进行管理，工作流程如图 4.11 所示，如果冷数据区有满足移入热数据区条件的缓存项，进行对应操作。

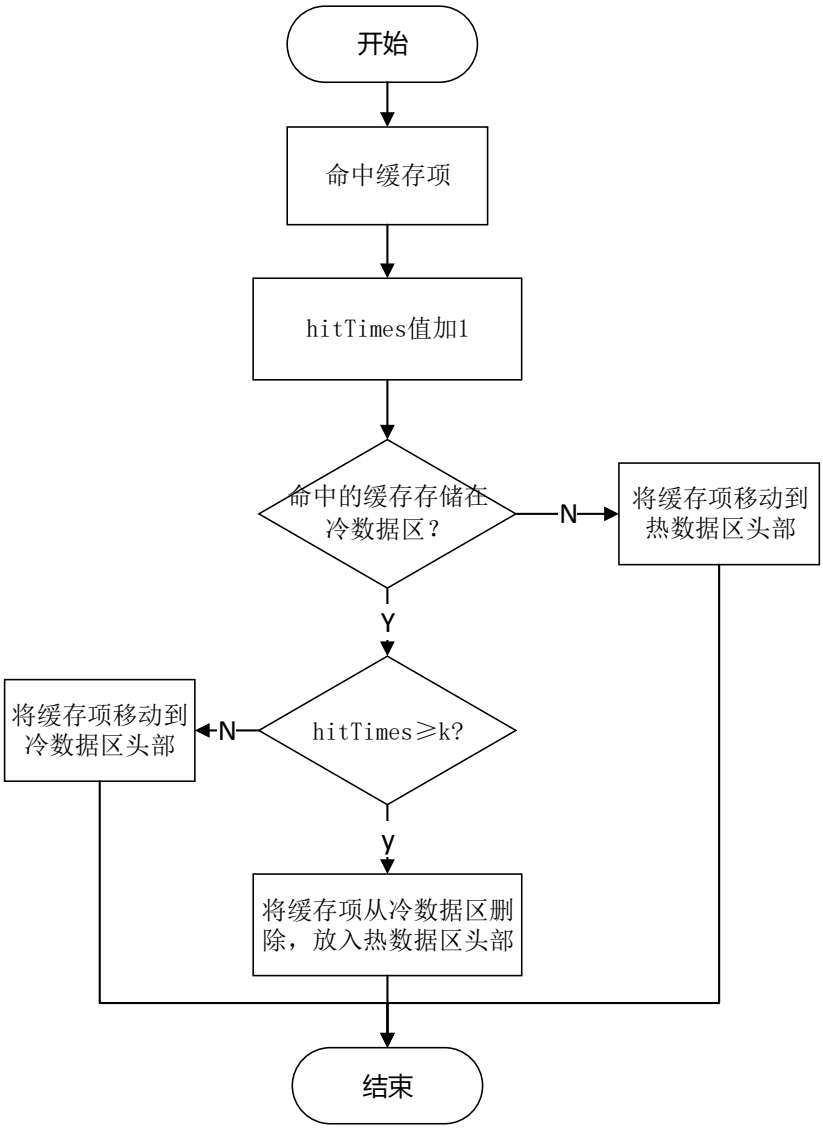



图4.11 缓存置换机制对缓存项的管理

### 4.2.4 一致性维护

当缓存项对应的数据表发生更新操作，会造成缓存与数据库数据不一致的情况，如果仍然使用旧的缓存，会读取到脏数据，本文设计了一种一致性策略来解决这个问题，保证用户读取到的是正确有效的缓存数据。

假设某缓存项已经存在，用 OPE=[$O_1$, $O_2$,..... $O_i$]表示用户对该缓存数据表的一系列操作。若 $O_i$ 为更新操作，即增删改操作，将 $O_i$ 收集到缓存项的 list 属性中，暂不连接数据执行 SQL 指令。若 $O_i$ 为查询操作，先判断 list 是否为空，如果 list 为空，表明在此之前没有进行更新，可以使用缓存；如果 list 不为空，则表明此次查询之前有更新操作，将 list 中的元素依次取出，连接数据库执行处理，直至 list 为空，最后再更新缓存数据。

这样，将多次更新操作收集起来一起批量执行，而不是每次更新都连接数据库并更新缓

存，减少了频繁与数据库建立连接的开销，节省了资源，提高了一致性维护的效率。一致性维护流程如图 4.12 所示。

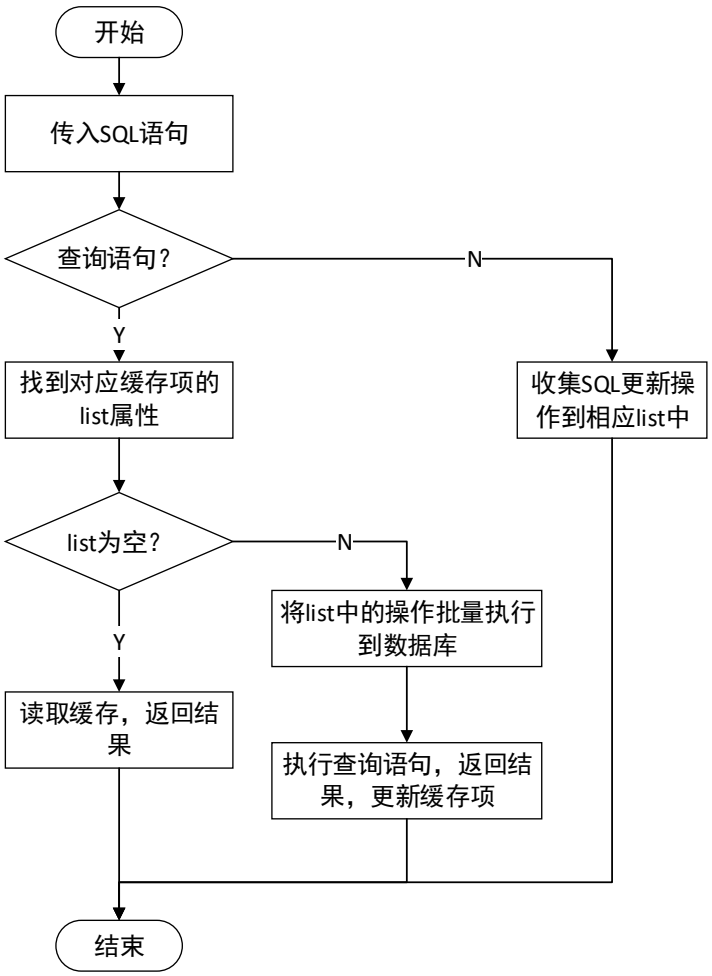



图4.12 一致性维护流程

## 4.3 测试分析

为了验证设计的 JDBC 缓存机制性能及改进的缓存置换策略对缓存命中率的影响，在本文设计的智能充电系统中进行了测试，测试分为命中率测试和响应时间测试两部分。后端业务程序部署在 tomcat 云服务器上，配置为 2 核 8G 的 64 位 Linux 系统，数据库为 2 核 4G 的阿里云 MySQL 数据库。

### 4.3.1 命中率测试

#### 4.3.1.1 测试方案

数据库中有 17 张表，其中超过 100 条数据的表有 6 张，超过 1000 条数据的表有 4 张，

超过 10000 条数据的表有 2 张，每条数据的大小约为 1K。实验时，随机生成 1000 条 SQL 指令，其中查询语句占 60%，增删改语句占 40%。

记录 JDBC 缓存使用 LRU 置换算法和本文置换算法在不同请求数目下的命中率。缓存命中的标准是，每次查询请求时，如果从缓存中取结果，记一次命中。连续请求时，总命中次数除以总请求次数就得到缓存命中率。

#### 4.3.1.2 测试数据

用连续请求次数作为横坐标，缓存命中率为纵坐标，得到图 4.13 中曲线。

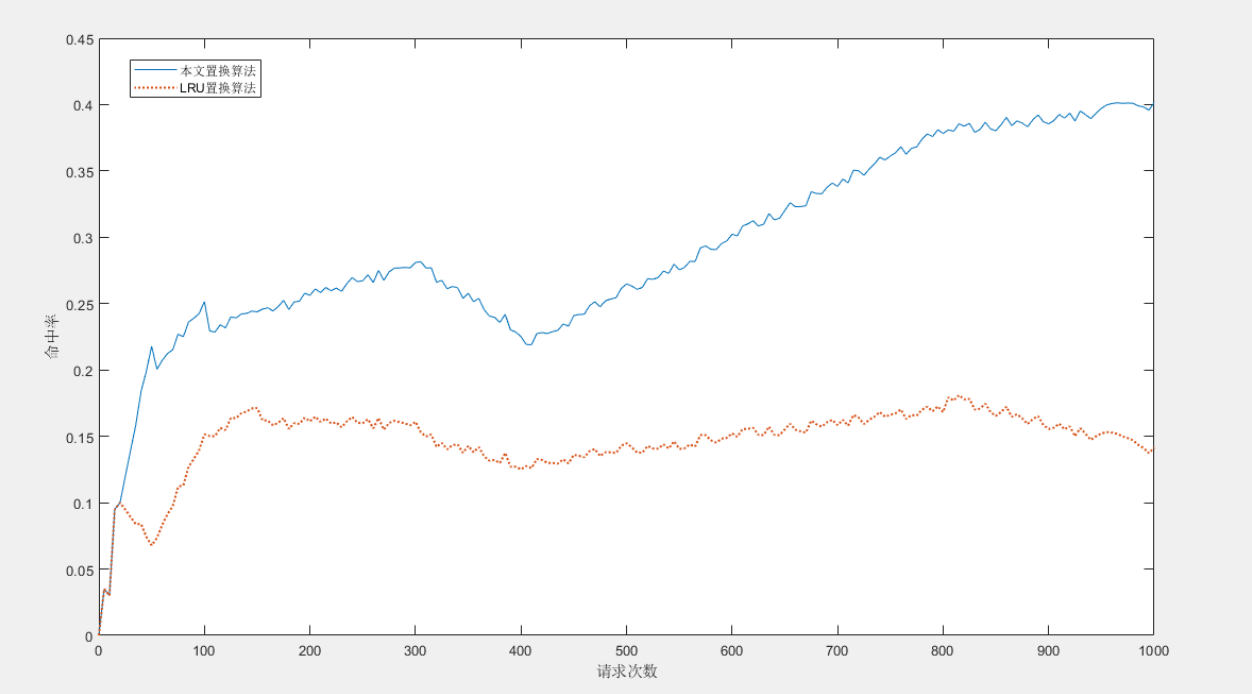



图4.13 缓存置换算法命中率对比曲线

#### 4.3.1.3 测试分析

从图 4.13 中我们可以看出，两条曲线随着请求次数的增加，缓存的结果集越来越多，缓存命中率都逐渐升高。曲线最后都趋于平稳，是因为缓存的空间有限，缓存的数据有上限。最终，LRU 算法命中率保持在 15%，本文算法命中率保持在 40%，比 LRU 算法高很多，具有更好的缓存性能。

### 4.3.2 响应时间测试

#### 4.3.2.1 测试方案

随机生成 1000 条 SQL 指令，查询语句占 60%，增删改语句占 40%，执行语句，对比未

使用 JDBC 缓存、使用 JDBC 缓存（基于 LRU 缓存置换算法）、使用 JDBC 缓存（基于本文缓存置换算法）三种情况下的平均响应时间。

#### 4.3.2.2 测试数据

用连续请求次数作为横坐标，平均响应时间为纵坐标，分别绘制未使用 JDBC 缓存、使用 JDBC 缓存（基于 LRU 缓存置换算法）、使用 JDBC 缓存（基于本文缓存置换算法）三种情况下的平均响应时间曲线，得到图 4.14。

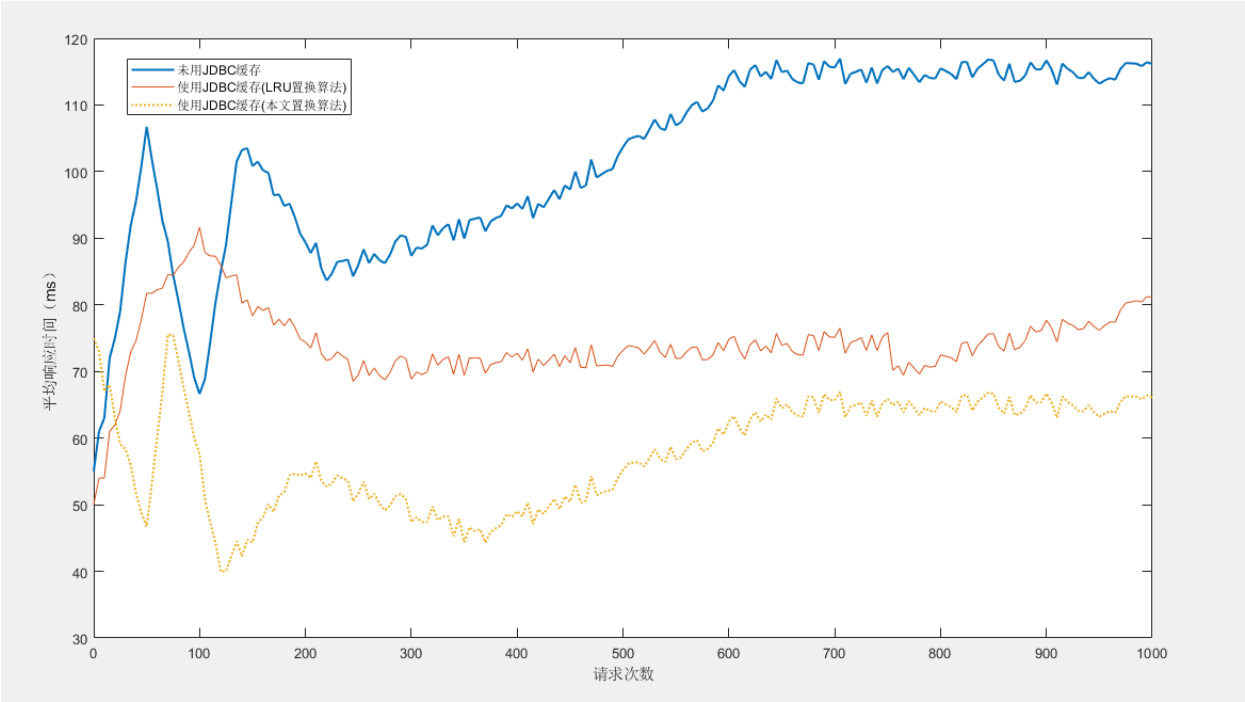



图4.14 平均响应时间对比

#### 4.3.2.3 测试分析

三条曲线整体的趋势一致，最开始因为请求次数少，缓存数据不规律，导致响应时间的波动大，随着请求次数的增多，缓存项经过多次管理和置换，处于一个比较规律的状态，曲线逐渐稳定。

随着查询次数的增多，缓存也在增多，越来越多的请求可以直接从缓存中读取结果，使用 JDBC 缓存方案比未使用缓存时花费时间少很多。而采用两种不同置换算法的 JDBC 缓存方案也存在差异，使用本文置换算法的 JDBC 缓存因为命中率更高，所以平均响应时间更短。

使用 LRU 置换算法的 JDBC 缓存方案响应时间比未使用 JDBC 缓存少了三分之一，使用本文置换算法的 JDBC 缓存方案响应时间比未使用 JDBC 缓存少了近一倍。

## 4.4 本章小结

本章先提出了系统高并发的问题，然后根据性能消耗原因引出服务器缓存，比较现有技术后选择较为轻便的 JDBC 缓存技术进行深入研究，指出现有 JDBC 缓存技术的不足并提出自己的改进方法，设计了符合智能充电系统的 JDBC 缓存方案。在设计时，先分析确定缓存对象特性和数据存储结构，修改扩展 JDBC API 接口，实现了对 SQL 查询指令的前置处理；然后重点研究缓存置换算法，提出一种改进的缓存置换策略，提高了缓存命中率；同时设计一致性维护策略保证缓存的准确。本文设计的 JDBC 缓存方案比现有 JDBC 缓存技术有更高的缓存命中率，整体缓存性能更好。

本章最后，进行模拟应用测试，实验表明，本文设计的 JDBC 缓存机制可以保持 40%的命中率，在 1000 条请求量下响应时间能够稳定缩短一倍，可以有效提高系统响应速度。

# 第五章 总结与展望

## 5.1 主要工作和结论

本文设计实现了电动自行车智能充电系统，为用户提供多种在线服务，帮助企业经营管理。同时，着重对系统性能进行了两个方面的优化：针对充电桩数据量大时搜索用户邻近充电桩速度慢的问题，设计一种查找算法，结合自主设计的 B-GeoHash 编码实现快速搜索；针对高峰时段数据操作高并发的情况，设计改进的 JDBC 缓存机制，提高了缓存命中率，减小了服务器连接数据库的开销，提升了系统响应速度。本文主要做了以下工作：

（1）调研电动自行车充电桩市场发展现状，总结当前充电服务系统存在的一些问题，并设计实现改进的智能充电系统。系统包含硬件系统、业务云、客户端三大部分。硬件系统中主控板具有过流、过载、短路保护功能，通信控制模块实现设备数据上报和平台指令下发。云平台中的设备云实现设备通信和设备数据存储管理，业务云处理业务逻辑，为客户端提供支撑。客户端开发分为移动用户端、移动运维端和 Web 管理平台三部分，为相关人员提供全方位服务。

（2）对用户服务的一个核心部分——邻近充电桩搜索服务进行优化。引入 GIS 系统中的空间查询，研究了现有空间索引机制，提出了一种基于四叉树网格划分的邻近充电桩快速搜索算法。该算法在搜索时划分的网格具有多级非均匀的特点，避免了数据冗余。对 GeoHash 编码进行改进，设计充电桩 B-GeoHash 编码，根据编码前缀包含的特性进行前缀匹配，实现快速筛选。

（3）从缓存入手对数据库操作进行优化，提升系统的并发处理能力。对 JDBC 接口进行扩展，设计存储结构，实现了简单查询请求结果集的缓存。当 SQL 请求对应的结果集缓存存在时，无需与数据库建立连接，直接使用缓存，加快了数据处理速度。同时，对内存空间进行合理分配，达到容量阈值时设计改进的置换策略进行缓存置换。当发生更新操作，导致从缓存中读取到脏数据时，设计一种高效的维护机制保证缓存数据一致。使用改进的 JDBC 缓存技术后，缓存保持 40%的命中率，操作响应时间缩短了一倍。

## 5.2 研究展望

本文设计了一套智能化的电动自行车充电系统，并对大数据量时邻近充电桩查询慢、重复数据库操作导致的响应速度慢两个问题进行优化，为用户提供功能完备、并发性能良好的充电服务，但依然存在不足之处，需要改进和完善，具体如下：

（1）系统的业务还有可扩展的地方，本系统虽然收集了电池的实时充电功率，但没有做

进一步的数据挖掘，后续可调研电动自行车电池充电模式，建立一些通用模型，对比功率曲线，为用户提供电池性能建议。

（2）邻近充电桩快速搜索算法中，采用地球球面公式计算用户与充电桩的距离，但这并不代表用户到达充电桩的实际里程，后续可以结合路网信息，为用户提供更精准的服务。

（3）JDBC 缓存机制仅对 SQL 语句做了简单处理，后续可考虑对不同的 SQL 语句进行深入解析，分析出关键信息，实现对带 WHERE 项的查询语句的缓存。

# 致谢

本设计的成功完成离不开导师陈科明副教授的认真指导。陈老师是一位优秀的导师，在自己的领域深入钻研，并注重理论与实践的结合。研究生期间，陈老师提供了内容丰富的学习平台，使我不仅可以钻研基础知识，也可以放眼市场，放眼未来，看到自己所做的东西的价值，从而更好地学习。写论文的过程中，陈老师一直严格要求，悉心指导，给我带来很大帮助，非常感谢陈老师。

同时，也感谢我的同学及同事们给予我的建议和指导，每当我遇到问题去找他们，他们都很耐心地讲解，与他们相处非常融洽、开心。

最后，感谢我亲爱的同门们陪我一起度过了愉快的研究生生涯，感谢我的男朋友一直陪伴我，在我写论文的紧张时候默默支持我。

# 参考文献


[1] 中国产业信息网.2018 年中国电动自行车产量、电动自行车保有量、各省市产量发展趋势分析[EB/OL]. [2019-11-04].http://www.chyxx.com/industry/201911/801292.html.

[2] 张宝山.依法治理让电动车走稳走好[J].中国人大,2019(15):46-47.

[3] 中华人民共和国公安部.关于规范电动车停放充电加强火灾防范的通告[EB/OL]. [2017-12-30].http://www.mps.gov.cn/n6557558/c5958191/content.html,2017-12-31.

[4] 于碧涵. 联网式电动自行车充电桩平台关键技术研究[D].杭州:浙江大学,2019:2-3.

[5] Kumar Dhaneesh,Krull Cornelius,Yin Yuefeng,Medhekar Nikhil V,Schiffrin Agustin. Electric Field Control of Molecular Charge State in a Single-Component 2D Organic Nanoarray.[J]. ACS nano,2019,13(10):22-24.

[6] 王锦尧.国内外电动自行车充电桩对比[J].合作经济与科技,2017(01):71-72.

[7] Fran Domazetović,Ante Šiljeg,Nina Lončar,Ivan Marić. GIS automated multicriteria analysis (GAMA) method for susceptibility modelling[J]. MethodsX,2019,10(02):45-47.

[8] Sehwa Park,Seog Park. Reverse collective spatial keyword query processing on road networks with G-tree index structure[J]. Information Systems,2019,84(04):33-42.

[9] Linjia Hu,Saeid Nooshabadi. High-dimensional image descriptor matching using highly parallel KD-tree construction and approximate nearest neighbor search[J]. Journal of Parallel and Distributed Computing,2019,132(12):13-15.

[10] Brinkhoff T , Kriegel H P , Seeger B . Parallel processing of spatial joins using R-trees[J]. Proc Acm Sigmod, 1993, 22(02):237-246.

[11] 史杏荣,孙贞寿,曹爱军.基于固定网格划分和面向类对象的四分树空间索引机制[J].小型微型计算机系统,1998(10):25-32.

[12] 陈敏,王晶海.R*-树空间索引的优化研究[J].计算机应用,2007(10):2581-2583.

[13] 张军旗,周向东,王梅,施伯乐.基于聚类分解的高维度量空间索引 B~+-Tree[J].软件学报,2008(06):1401-1412.

[14] 申丹丹.一种 HBase 空间索引设计[J].信息与电脑(理论版),2016(08):51-52.

[15] Ken Alabi. Digital blockchain networks appear to be following Metcalfe’s Law[J]. Electronic Commerce Research and Applications,2017,24(11):27-29.

[16] David Rozado,Ahmad El Shoghri,Raja Jurdak. Gaze dependant prefetching of web content to increase speed and comfort of web browsing[J]. International Journal of Human - Computer Studies,2015,78(08):66-72.

[17] 张敏.探究互联网缓存机制[J].现代工业经济和信息化,2019,9(05):73-74.

[18] Zarepour E, Karim A, Sharma A, et al. Characterising Power Saving for Device-to-Device Browser Cache Cooperation[J].Journal of Network and Computer Applications,2016,

67(C):118-127.

[19] Wu Z, Lu Z, Zhang W, et al. A data-driven approach of performance evaluation for cache server groups in content delivery network[J]. Journal of Parallel and Distributed Computing,2018, 119:162-171.

[20] 安仲奇,杜昊,李强,霍志刚,马捷.基于高性能I/O技术的Memcached优化研究[J].计算机研究与发展,2018,55(04):864-874.

[21] Zheng R, Liu Q,Jin H. Memory Data Management System for Rendering Applications[C].2015 Second International Conference on Mathematics and Computers in Sciences and in Industry (MCSI). IEEE, 2015.

[22] Anne N.Gade,Tine S.Larsen.Nissen,Rasmus L.Jensen. REDIS: A value-based decision support tool for renovation of building portfolios[J]. Building and Environment,2018,142(04):55-57.

[23] 郎泓钰,任永功.基于 Redis 内存数据库的快速查找算法[J].计算机应用与软件,2016,33(05):40-43.

[24] Meer M V D,Kurthnelson Z,Redish A D.Information Processing in Decision-Making Systems[J]. 2012, 18(4):342-59.

[25] 江泽源,刘辉林,吴刚,王国仁.内存数据库的可用性综述[J].华东师范大学学报(自然科学版),2014(05):82-88.

[26] Information Technology; Studies Conducted by C. Li et al on Information Technology Recently Reported (FluteDB: An efficient and scalable in-memory time series database for sensor-cloud)[J]. Computers, Networks & Communications,2018(45):112-115.

[27] 罗建明.基于 SSH 工资查询系统的研究[J].信息与电脑(理论版),2019(09):99-100.

[28] 杨静,彭荣华,赵世民,钟阳万.基于 OneNET 物联网平台的车位锁控制系统[J].科技与创新,2019(08):94-95.

[29] Peter Mersky. Grumman F-14 Tomcat Owners' Workshop Manual, All Models 1970-2006[J]. Naval Aviation News,2019,101(01):24-25.

[30] 丁亚争,王琬茹.基于 JDBC 的数据库连接池技术研究[J].科技信息,2009(04):500.

[31] Ivan Merelli,Federico Fornari,Fabio Tordini,Daniele D'Agostino,Marco Aldinucci,Daniele Cesini. Exploiting Docker containers over Grid computing for a comprehensive study of chromatin conformation in different cell types[J]. Journal of Parallel and Distributed Computing,2019,134(22):43.

[32] 熊耀. RESTful Web 服务在云平台下的设计与实现[D].成都:电子科技大学,2018:31.

[33] 刘梓良.面向 Java Web 的 3A 安全框架研究与设计[D].西安:西安电子科技大学,2017:45-52.

[34] 钟俊林.基于微服务架构的自助微商城的研究与实现[D].北京:北京邮电大学,2019:5-6.

[35] 梁伟强.电动汽车充电服务平台研究与开发[D].北京:华北电力大学,2017:11.

[36] 王鹏强.基于 vue 的 MVVM 框架的研究与分析[J].电脑知识与技术,2019,15(11):97-98.

[37]陈岩.轻量级响应式框架 Vue.js 应用分析[J].中国管理信息化,2018,21(03):181-183.

[38]冯传波,彭章友,张钟浩.基于 Vue.js 的移动应用可视化平台的研究[J].工业控制计算机,2019,32(05):102-103.

[39]Jahanian Ali,Keshvari Shaiyan,Rosenholtz Ruth. Web pages: What can you see in a single fixation?[J]. Cognitive research: principles and implications,2018,3(1) :213-222.

[40]刘冬季.浅析 GIS 与视频集成技术的创新应用[J].中国安防,2014(10):58-60.

[41]张新长,曾广鸿,张青年.城市地理信息系统[M].北京:科学出版社,2001:8-10.

[42]董鹏.分布式空间信息的高效查询与分析系统研究[D].北京:中国科学院研究生院（遥感应用研究所）,2003:23.

[43]刘彬.气象 GIS 空间数据集成组织与系统原型设计[D].南京:南京信息工程大学,2016:25.

[44]刘佳奇,刘勇,边少锋.斜轴墨卡托投影在世界地图表达中的应用[J].测绘科学,2019,44(11):103-108.

[45]Zhao Bingxu,Wang Yingjie,Li Yingshu,Gao Yang,Tong Xiangrong. Task Allocation Model Based on Worker Friend Relationship for Mobile Crowdsourcing.[J]. Sensors (Basel, Switzerland),2019,19(4) :33-36.

[46]刘家铭.基于 J2EE 的师生教学娱乐一体化系统的设计与实现[J].现代信息科技,2019,3(18):18-20.

[47]Han M, Yu S, Baek W. Secure and Dynamic Core and Cache Partitioning for Safe and Efficient Server Consolidation[C].2018.

[48]Smiianov V A, Dryha N O, Smiianova O I, et al. Development of informational-communicative system, created to improve medical help for family medicine doctors[J]. Wiadomości lekarskie (Warsaw, Poland: 1960), 2018, 71(2):331-334.

[49]梁珂.电动汽车充电桩移动监控与故障诊断系统研究[D].青岛:青岛科技大学,2019:52.

[50]方是源.基于 Java 语言实现数据库的访问研究[J].电子技术与软件工程,2017(16):184.

[51]欧阳宏基,葛萌,陈伟.基于 JDBC 的数据持久化层性能优化研究[J].网络新媒体技术,2016,5(05):9-15.

# 附录

## 作者在读期间发表的学术论文及参加的科研项目

### 科研项目

［1］***远程运维系统研究设计，横向课题

［2］***国际物流系统研究设计，横向课题

### 科研获奖

［1］第十四届研究生电子设计竞赛　华东赛区一等奖