gStore-Music 数据查询

2019 年 8 月 10 日

一、 实验环境

1. 实验工具

Java11、Spring-boot 2.1.7、thymeleaf 2.1.7、gson2.8.5、Tomcat 9.0.22、gStore 等

2. 实验平台

Windows 10、CPU i7-8700、Mem 16 G、IDEA 2018

3. 实验数据

使用 Music 数据集

二、 实验数据

1. 安装 gStore

- 安装环境：CentOS7、Java11、git
- 安装步骤：
  - 下载 gStore，执行 git clone https://github.com/pkumod/gStore.git
  - 在 gStore 目录下执行./scripts/setup/setup_centos.sh
  - 在 gStore 目录下执行 make pre
  - 在 gStore 目录下执行 make

2. 导入 music 数据

- 在 gStore/data 下面创建 music 文件夹，并将 music.nt 文件复制到 music 文件夹下
- 在 gStore 目录下执行 bin/build music data/music/music.nt
- 在 gStore 目录下执行 bin/ghttp music 9000，在浏览器输入网址 127.0.0.1:9000/admin.html，可看到如图 1 的页面。

三、 实验过程

在 Java11 环境下，利用 IDEA2018 工具创建 Spring-boot 项目，使用 maven 构建该项目，同时使用 thymeleaf 模板。项目目录结构如图 2 所示。

1. 创建对象实体

本项目在 model 下创建了 3 个实体类 DBInfo、Entry、Music。

- DBInfo 用于记录 gStore 数据库信息
- Entry 用于记录 Json 格式查询的最小单元实体{type, value}
- Music 用于记录 RDF 三元组<subject, predicate, object>

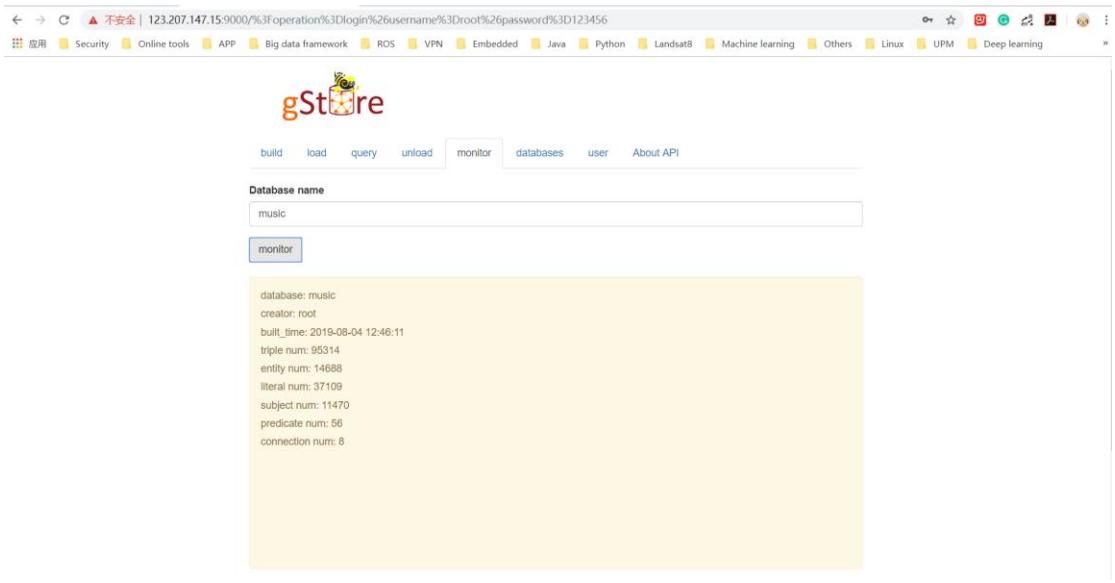

图 1 ghttp 示例页面

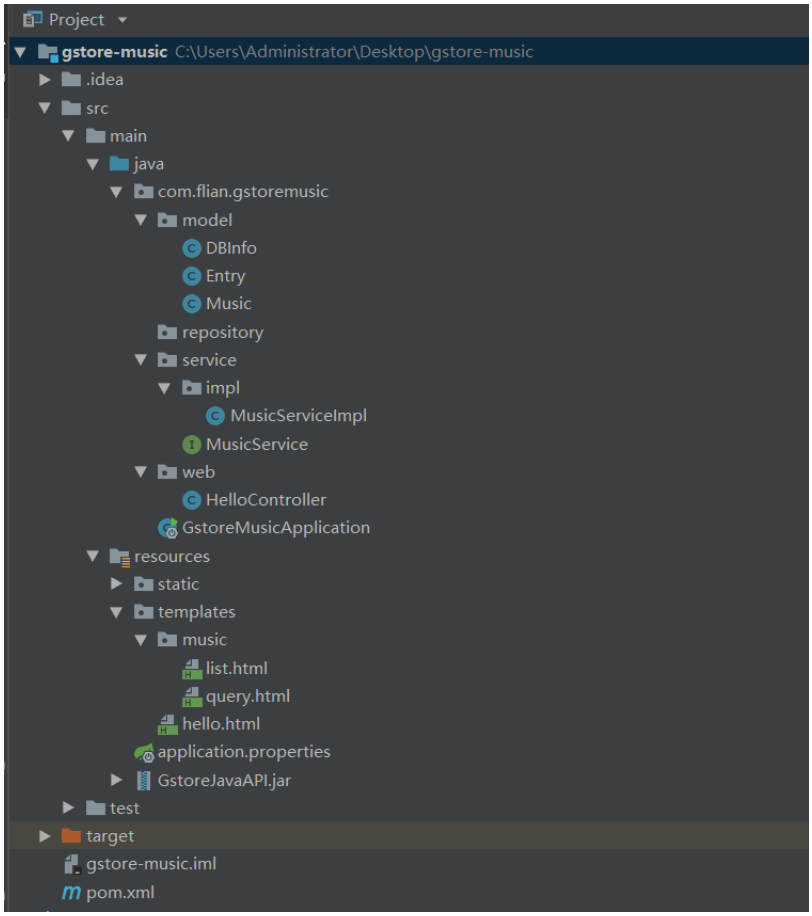

图 2 项目结构

2. 定义 service

在 service 下定义了 MusicService 接口类，包含如下 3 个方法：

```
public interface MusicService {
    public List<Music> getAllMusic(int pageNum, int pageSize) throws
Exception;
    public Map<String, List<Map<String,Entry>>>
findMusicBySparql(String sparql, int pageNum, int pageSize);
    public DBInfo getDBInfo(String DBName);
}
```

在其实现类中创建 gStore 连接：

```
    // connect to gStore
GstoreConnector gc = new GstoreConnector("127.0.0.1", 9000,
"root", "123456");
```

其实现类如下：

- 查询数据库信息

```
    @Override
    public DBInfo getDBInfo(String DBName) {
        String res = gc.monitor(DBName);
        if (res.isEmpty())
            return null;
        return new Gson().fromJson(res, new TypeToken<DBInfo>()
{}.getType());
    }
```

- 按<subject, predicate, object>三元组返回所有数据库的数据，这里利用 LIMIT 和 OFFSET 实现了分页查询

```
    @Override
    public List<Music> getAllMusic(int pageNum, int pageSize) throws
Exception{
        // 查询所有数据
        String sparql= String .format("SELECT * WHERE {?s ?p ?o}
LIMIT %s OFFSET %s", pageSize, (pageNum-1)*pageSize);
        String res = gc.query("music", "json", sparql);
        // 解析查询结果
        JsonElement jsonElement = new JsonParser().parse(res);
        if (jsonElement.isJsonNull()) {
            return null;
        } else if (jsonElement.isJsonObject()) {
```

```
            JsonElement results =
jsonElement.getAsJsonObject().get("results").getAsJsonObject().get("bin
dings");
            List<Music> MusicList = gson.fromJson(results, new
TypeToken<List<Music>>() {}.getType());
            for (int i=0; i < MusicList.size(); i++)
                MusicList.get(i).setId(i);
            return MusicList;
        } else if (jsonElement.isJsonArray()) {
            return null;
        } else if (jsonElement.isJsonPrimitive()) {
            return null;
        }
        return null;
    }
```

- 根据给定的 sparql，返回查询结果。其中，若 sparql 中无 LIMIT，则添加 LIMIT 和 OFFSET 来实现分页查询。

```
    @Override
    public Map<String, List<Map<String,Entry>>>
findMusicBySparql(String sparql, int pageNum, int pageSize) {
        String res = gc.query("music", "json", sparql);

        Map<String, List<Map<String,Entry>>> result = new
LinkedHashMap<String, List<Map<String,Entry>>>();
        ArrayList<Map<String,Entry>> status = new
ArrayList<Map<String,Entry>>();
        List<Map<String,Entry>> musics = new
ArrayList<Map<String,Entry>>();
        // 解析查询结果
        JsonElement jsonElement = new JsonParser().parse(res);
        if (jsonElement.isJsonNull()) {
            return null;
        } else if (jsonElement.isJsonObject()) {
            String statusCode =
jsonElement.getAsJsonObject().get("StatusCode").getAsString();
            String statusMsg =
jsonElement.getAsJsonObject().get("StatusMsg").getAsString();

            Map<String,Entry> tempM = new LinkedHashMap<String,
Entry>();
```

```
            tempM.put("statusCode", new Entry("statusCode",
statusCode));
            tempM.put("statusMsg", new Entry("statusMsg", statusMsg));
            status.add(tempM);
            result.put("status", status);

            // error
            if(!statusCode.equals("0")) {
                result.put("musicList", musics);
                return result;
            }

            JsonArray attName =
jsonElement.getAsJsonObject().get("head").getAsJsonObject().get("vars")
.getAsJsonArray();
            JsonArray results =
jsonElement.getAsJsonObject().get("results").getAsJsonObject().get("bin
dings").getAsJsonArray();
            for (JsonElement music:results) {
                Map<String,Entry> tempMusic = new LinkedHashMap<String,
Entry>();
                for (JsonElement att:attName) {
                    JsonObject item =
music.getAsJsonObject().get(att.getAsString()).getAsJsonObject();
                    tempMusic.put(att.getAsString(), new
Entry(item.get("type").getAsString(),
item.get("value").getAsString()));
                }
                musics.add(tempMusic);
            }
            result.put("musicList", musics);
            return result;
            // List<Music> MusicList = gson.fromJson(results, new
TypeToken<List<Map<String,Entry>>>() {}.getType());
        } else if (jsonElement.isJsonArray()) {
            return null;
        } else if (jsonElement.isJsonPrimitive()) {
            return null;
        }
        return null;
    }
```

3. 创建 controller，定义了 `bySparql` 接口

```
@Controller
public class HelloController {

    MusicService musicService = new MusicServiceImpl();

    @RequestMapping("/")
    public String index() {
        return "redirect:/bySparql";
    }

    @RequestMapping("/list")
    public String list(Model model,
                       @RequestParam(required =
false,defaultValue="1",value="pageNum") Integer pageNum,

@RequestParam(defaultValue="10",value="pageSize") Integer pageSize)
throws Exception {
        if(pageNum == null || pageNum <= 0) {
            pageNum = 1;
        }
        if(pageSize == null) {
            pageSize = 10;
        }
        List<Music> musics = musicService.getAllMusic(pageNum,
pageSize);
        model.addAttribute("musics", musics);

        DBInfo db = musicService.getDBInfo("music");
        model.addAttribute("dbInfo", db);
        model.addAttribute("pageNum", pageNum);
        model.addAttribute("pageSize", pageSize);
        int[] navigatePageNum = new int[5];
        int base = (pageNum-1)/5;
        int totalPages = (int) ((db.getTriple_num()+pageSize-
1)/pageSize);
        model.addAttribute("totalPages", totalPages);
        for(int i=0; i <5; i++) {
            int temp = base*5 + i + 1;
            if(temp <= totalPages)
                navigatePageNum[i] = temp;
            else
                break;
```

```
        }
        model.addAttribute("navigatePageNum", navigatePageNum);
        return "music/list";
    }

    @RequestMapping("/bySparql")
    public String bySparql(Model model,
                           String sparql,
                           @RequestParam(required =
false,defaultValue="1",value="pageNum") Integer pageNum,

@RequestParam(defaultValue="10",value="pageSize") Integer pageSize)
throws Exception {
        if(pageNum == null || pageNum <= 0) {
            pageNum = 1;
        }
        if(pageSize == null) {
            pageSize = 10;
        }
        model.addAttribute("sparql", sparql);

        if(sparql == null|| sparql.isEmpty())
            sparql= String .format("SELECT * WHERE {?s ?p ?o} LIMIT %s
OFFSET %s", pageSize, (pageNum-1)*pageSize);
        else if (sparql.toUpperCase().indexOf("LIMIT")==-1)
            sparql = sparql + " LIMIT " + pageSize + " OFFSET " +
(pageNum-1)*pageSize;
        Map<String, List<Map<String,Entry>>> musics =
musicService.findMusicBySparql(sparql, pageNum, pageSize);
        model.addAttribute("result", musics);
        DBInfo db = musicService.getDBInfo("music");
        model.addAttribute("dbInfo", db);

        model.addAttribute("pageNum", pageNum);
        model.addAttribute("pageSize", pageSize);
        int[] navigatePageNum = new int[5];
        int base = (pageNum-1)/5;
        int totalPages = (int) ((db.getTriple_num()+pageSize-
1)/pageSize);
        model.addAttribute("totalPages", totalPages);
        for(int i=0; i <5; i++) {
            int temp = base*5 + i + 1;
```

```
            if(temp <= totalPages)
                navigatePageNum[i] = temp;
            else
                break;
        }
        model.addAttribute("navigatePageNum", navigatePageNum);
        return "music/query";
    }
}
```

4. 创建页面，页面功能包含显示数据库信息，sparql 自定义查询和数据库信息显示。分页显示模块如下：

```
    <div class="modal-footer no-margin-top"
th:if="${result.get('status').get(0).get('statusCode').getValue().equal
s('0')}">
        <ul class="pagination pull-right no-margin">
            <li>
                <a th:href="'/list?pageNum=1'">First</a>
            </li>
            <li class="prev">
                <a th:href="'/list?pageNum=' + (${pageNum}-1)">
                    <i class="ace-icon fa fa-angle-double-left"></i>
                </a>
            </li>
            <li th:each="nav:${navigatePageNum}">
                <a th:href="'/list?pageNum='+${nav}" th:text="${nav}"
th:if="${nav != pageNum && nav!=0}"></a>
                <span style="font-weight: bold;background: #6FAED9;"
th:if="${nav == pageNum && nav!=0}" th:text="${nav}" ></span>
            </li>
            <li class="next">
                <a th:href="'/list?pageNum=' + (${pageNum}+1)">
                    <i class="ace-icon fa fa-angle-double-right"></i>
                </a>
            </li>

            <li>
                <a th:href="'/list?pageNum=' + ${totalPages}">Last</a>
            </li>
        </ul>
    </div>
```

此外，由于表格内容过长，无法在一行内全部显示。故将超出的内容隐藏，当鼠标悬浮在表格上方时，将显示全部内容，如图 3 所示。代码如下：

```
    <style>
        .td1 {
            white-space: nowrap;
            text-overflow: ellipsis;
            overflow: hidden;
        }
        .td2 {
            word-wrap: break-word   ;
        }
    </style>
```

```
        <tr th:each="music : ${result.get('musicList')}">
            <td th:each="item : ${music}"
th:text="${item.getValue().getValue()}" class="td1"
onmouseover="this.className='td2'"
onmouseout="this.className='td1'">Flian</td>
        </tr>
```

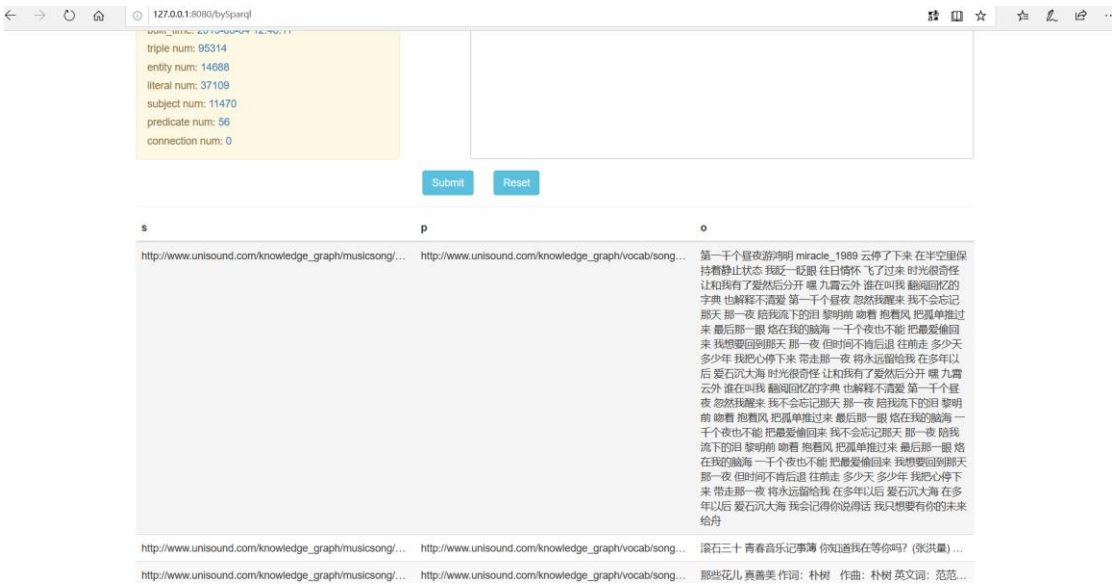

图 3 表格显示示例

四、 页面介绍

1. 软件运行

- 开启 gStore 服务，在 gStore 目录下执行 bin/ghttp music 9000
- 运行系统代码

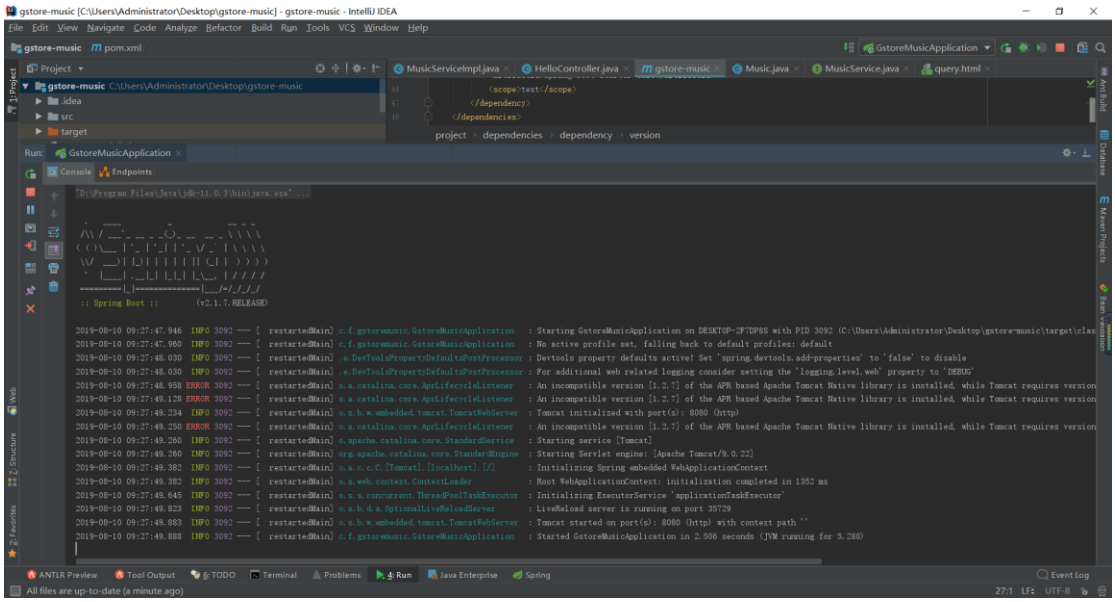

图 4 运行系统

2. 软件使用

- 在浏览器访问 http://127.0.0.1:9000/，界面显示如图 5 所示。左上方黄色区域显示了 Music 数据库的信息，右上方为 sparql 自定义查询，页面下方为 Music 数据库所有<s,p,o>数据(select * where {?s ?p ?o})。

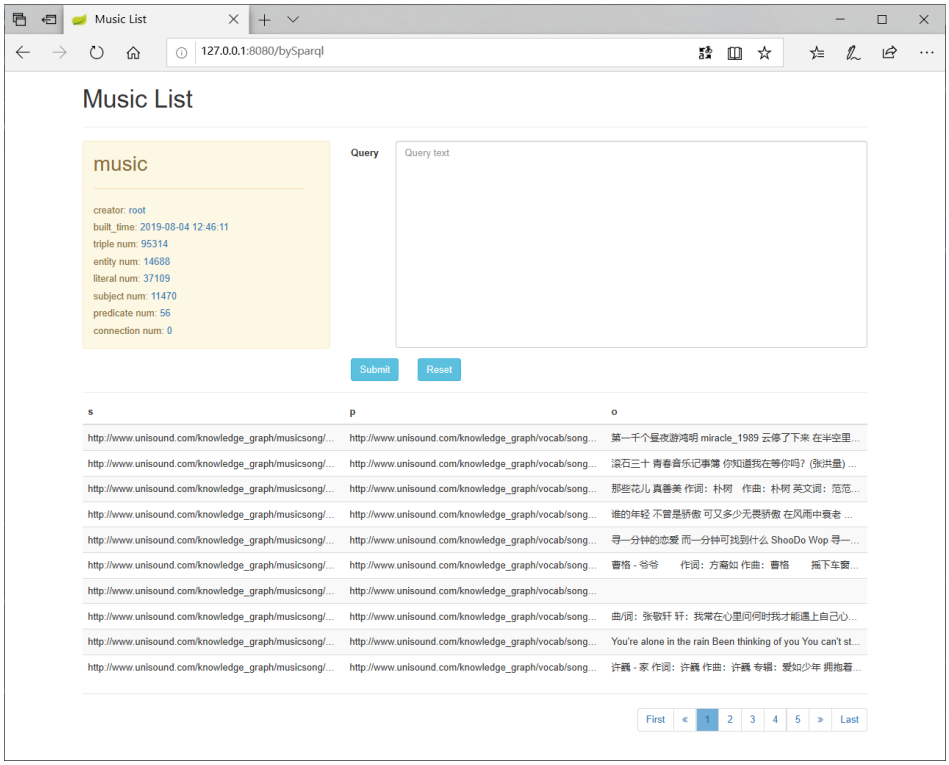

图 5 软件首页

- Sparql 查询演示，图 6 为错误查询，图 7 为正确查询(select ?p where

{?s ?p ?o})。

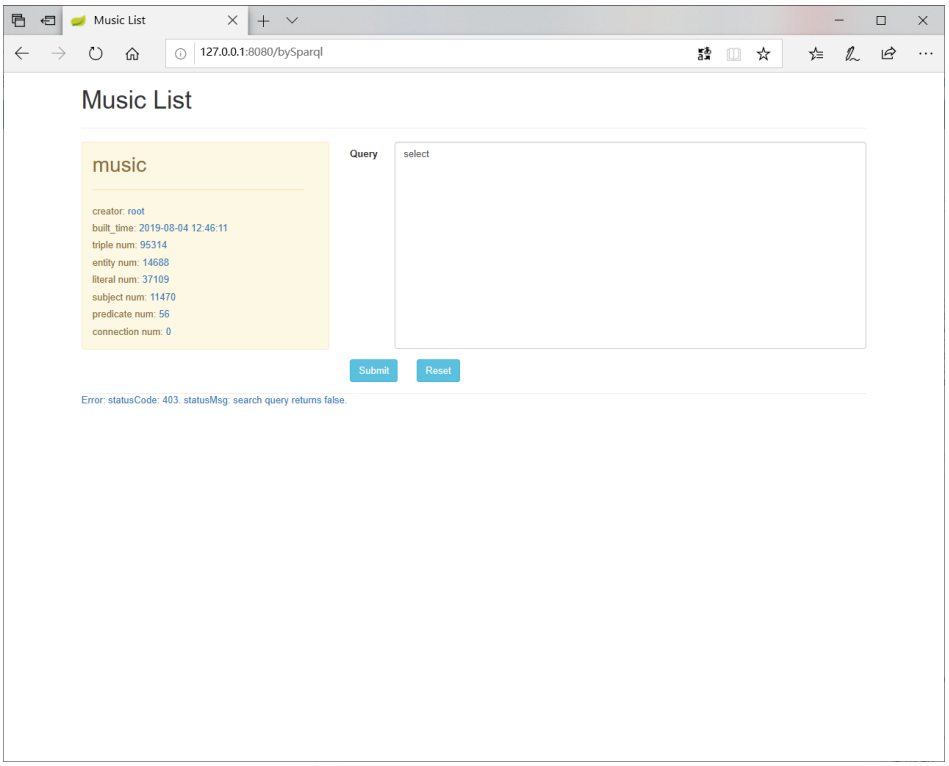

图 6 查询错误示例

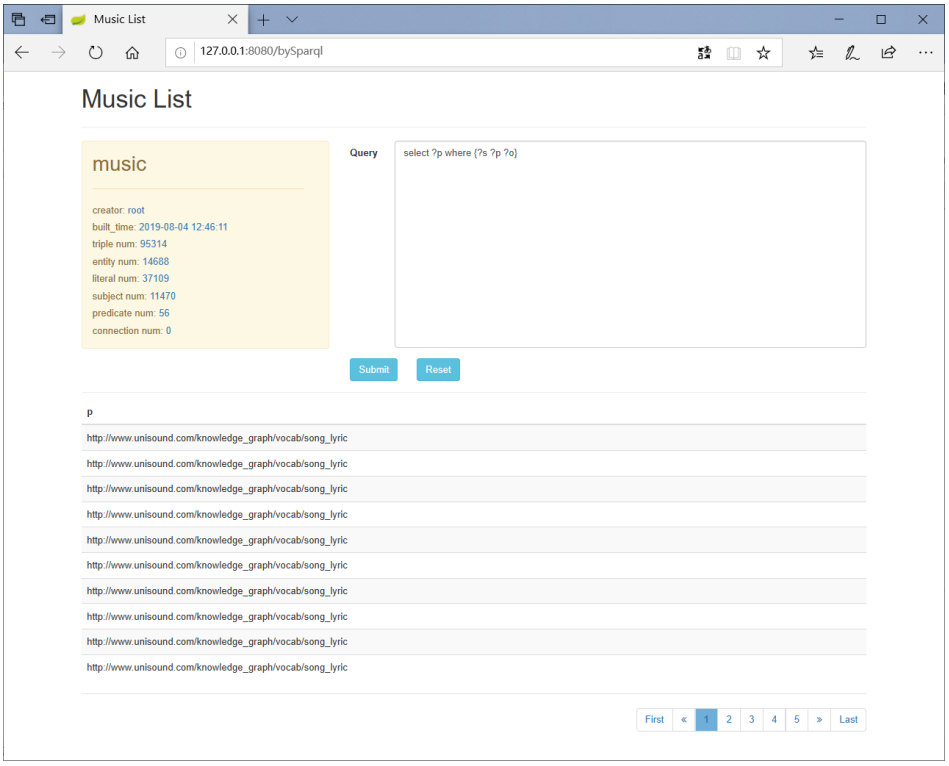

图 7 查询正确示例

- 页面特性显示，图 8 为鼠标悬浮显示全部内容，图 9 为分页显示。

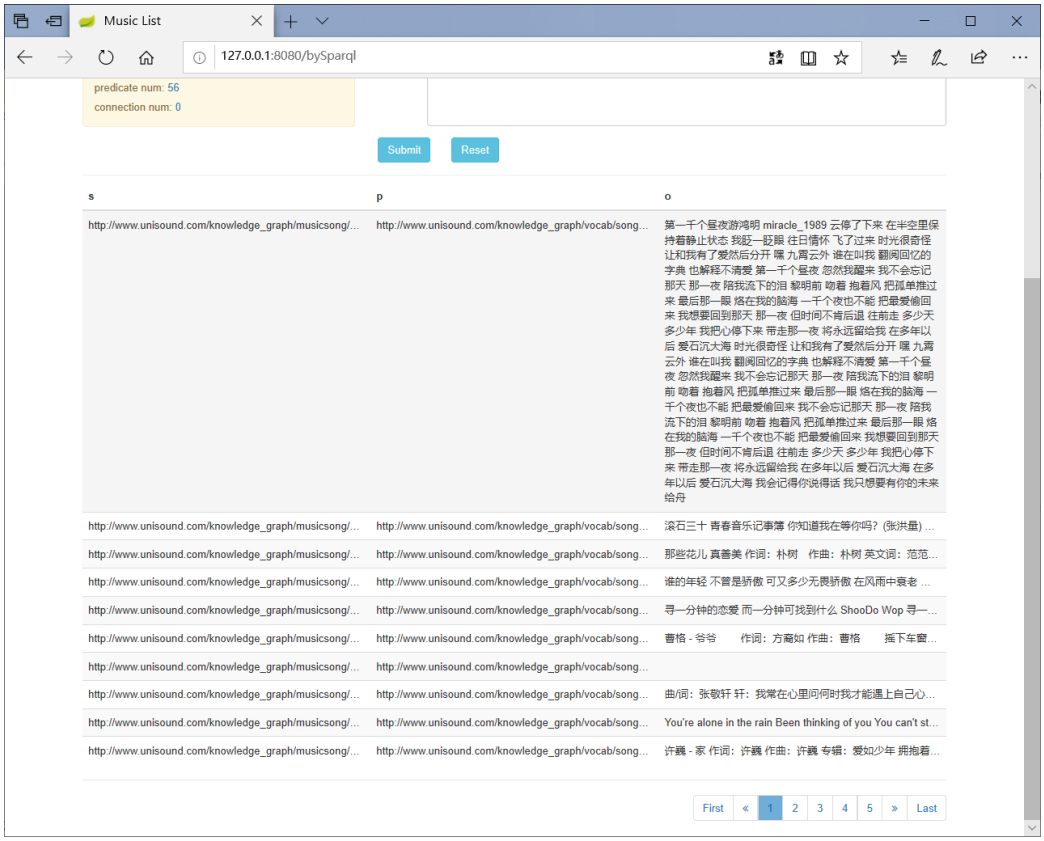

图 8 鼠标悬浮全部显示

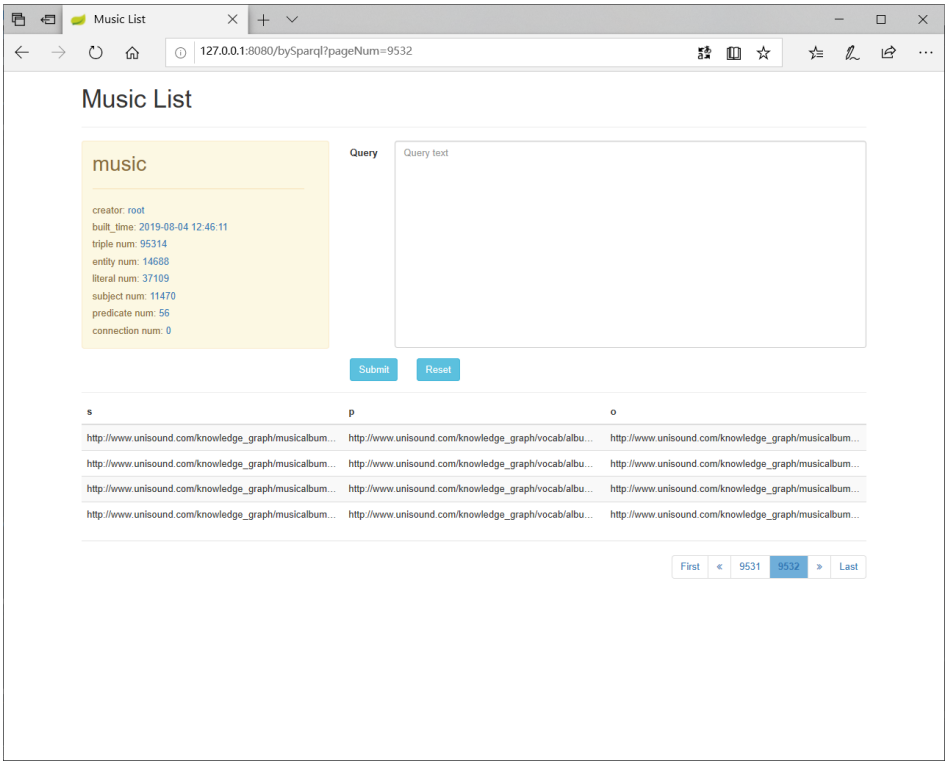

图 9 分页显示