纯真 IP 数据库格式详解

基本结构

QQWry.dat 文件在结构上分为 3 块：文件头，记录区，索引区。一般我们要查找 IP 时，先在索引区查找记录偏移，然后再到记录区读出信息。由于记录区的记录是不定长的，所以直接在记录区中搜索是不可能的。由于记录数比较多，如果我们遍历索引区也会是有点慢的，一般来说，我们可以用二分查找法搜索索引区，其速度比遍历索引区快若干数量级。图 1 是 QQWry.dat 的文件结构图。

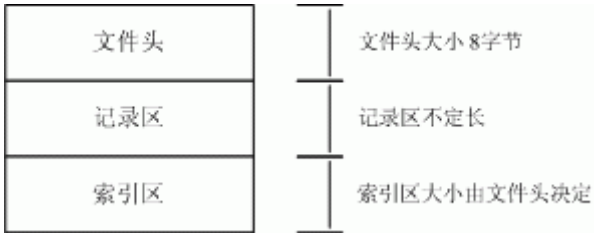

图 1. QQWry.dat 文件结构

要注意的是，QQWry.dat 里面全部采用了 little-endian 字节序

一．了解文件头

QQWry.dat 的文件头只有 8 个字节，其结构非常简单，首四个字节是第一条索引的绝对偏移，后四个字节是最后一条索引的绝对偏移。

二．了解记录区

每条 IP 记录都由国家和地区名组成，国家地区在这里并不是太确切，因为可能会查出来“清华大学计算机系”之类的，这里清华大学就成了国家名了，所以这个国家地区名和 IP 数据库制作的时候有关系。所以记录的格式有点像 QName，有一个全局部分和局部部分组成，我们这里还是沿用国家名和地区名的说法。

于是我们想象着一条记录的格式应该是：[IP 地址][国家名][地区名]，当然，这个没有什么问题，但是这只是最简单的情况。很显然，国家名和地区名可能会有很多的重复，如果每条记录都保存一个完整的名称拷贝是非常不理想的，所以我们就需要重定向以节省空间。所以为了得到一个国家名或者地区名，我们就有了两个可能：第一就是直接的字符串表示的国家名，第二就是一个 4 字节的结构，第一个字节表明了重定向的模式，后面 3 个字节是国家名或者地区名的实际偏移位置。对于国家名来说，情况还可能更复杂些，因为这样的重定向最多可能有两次。

那么什么是重定向模式？根据上面所说，一条记录的格式是[IP 地址][国家记录][地区记录]，如果国家记录是重定向的话，那么地区记录是有可能没有的，于是就有了两种情况，我管他叫做模式 1 和模式 2。我们对这些格式的情况举图说明：

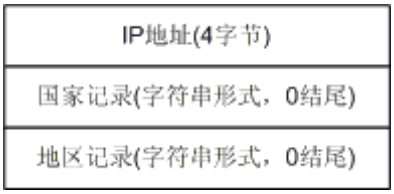

图 2. IP 记录的最简单形式

图 2 表示了最简单的 IP 记录格式，我想没有什么可以解释的

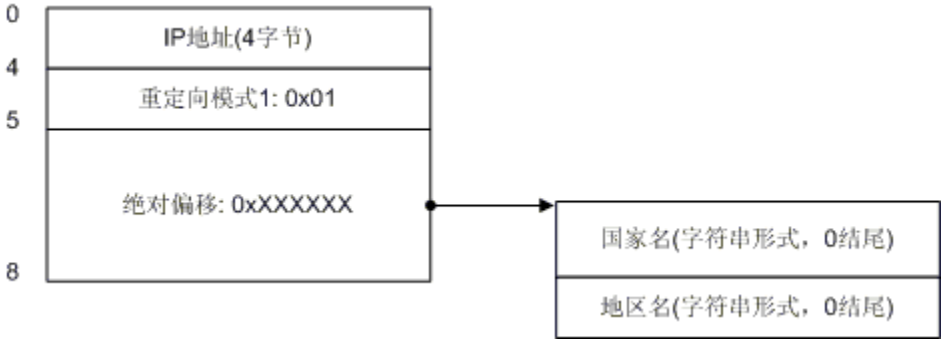

图 3. 重定向模式 1

图 3 演示了重定向模式 1 的情况。我们看到在模式 1 的情况下，地区记录也跟着国家记录走了，在 IP 地址之后只剩下了国家记录的 4 字节，后面 3 个字节构成了一个指针，指向了实际的国家名，然后又跟着地址名。模式 1 的标识字节是 0x01。

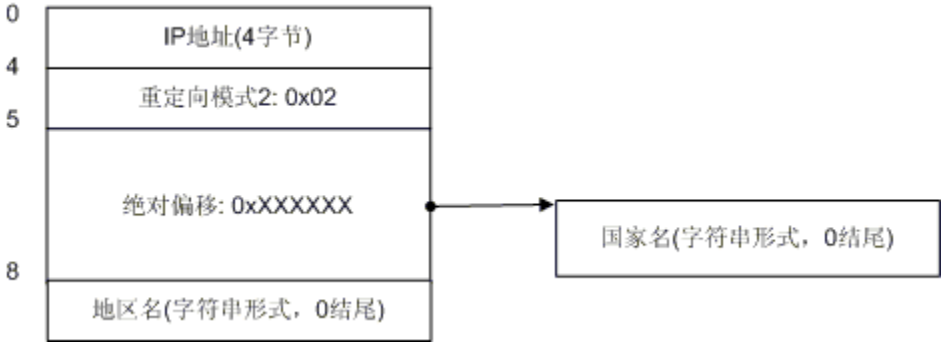

图 4. 重定向模式 2

图 4 演示了重定向模式 2 的情况。我们看到了在模式 2 的情况下（其标识字节是 0x02），地区记录没有跟着国家记录走，因此在国家记录之后 4 个字节之后还是有地区记录。我想你已经明白了模式 1 和模式 2 的区别，即：模式 1 的国家记录后面不会再有地区记录，模式 2 的国家记录后会有地区记录。下面我们来看一下更复杂的情况。

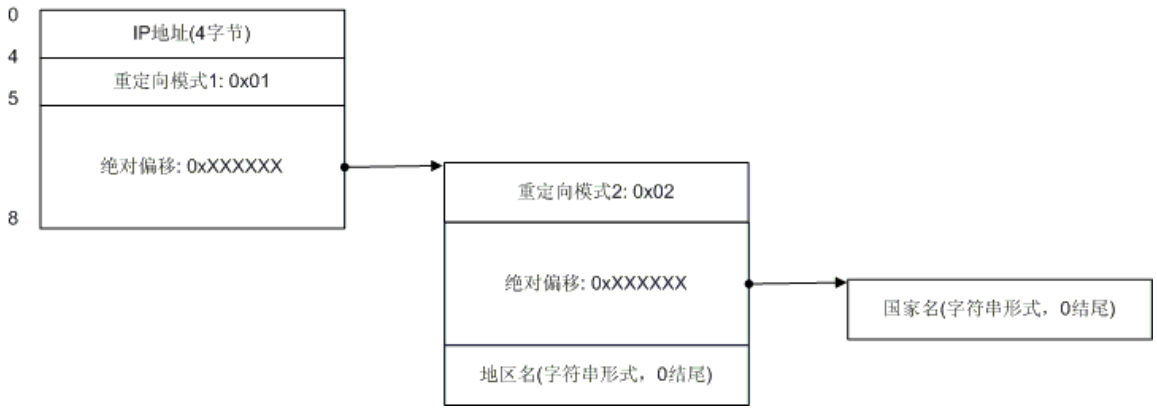

图 5．混和情况 1

图 5 演示了当国家记录为模式 1 的时候可能出现的更复杂情况，在这种情况下，重定向指向的位置仍然是个重定向，不过第二次重定向为模式 2。大家不用担心，没有模式 3 了，这个重定向也最多只有两次，并且如果发生了第二次重定向，则其一定为模式 2，而且这种情况只会发生在国家记录上，对于地区记录，模式 1 和模式 2 是一样的，地区记录也不会发生 2 次重定向。不过，这个图还可以更复杂，如图 7：

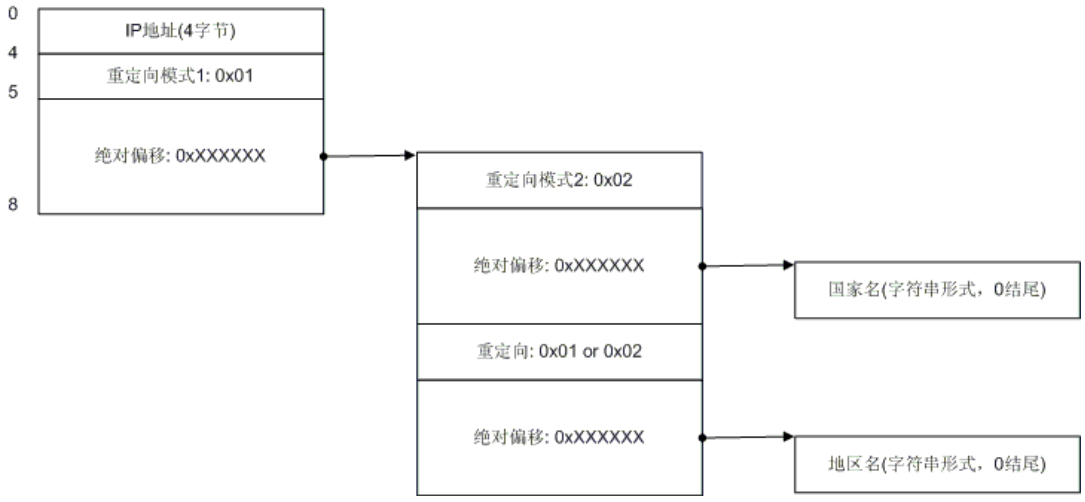

图 6．混和情况 2

图 6 是模式 1 下最复杂的混和情况，不过我想应该也很好理解，只不过地区记录也来重定向而已，有一点我要提醒你，如果重定向的地址是 0，则表示未知的地区名。

所以我们总结如下：一条 IP 记录由[IP 地址][国家记录][地区记录]组成，对于国家记录，可以有三种表示方式：字符串形式，重定向模式 1 和重定向模式 2。对于地区记录，可以有两种表示方式：字符串形式和重定向，另外有一条规则：

重定向模式 1 的国家记录后不能跟地区记录。按照这个总结，在这些方式中合理组合，就构成了 IP 记录的所有可能情况。

设计的理由

在我们继续去了解索引区的结构之前，我们先来了解一下为何记录区的结构要如此设计。我想你可能想到了答案：字符串重用。没错，在这种结构下，对于一个国家名和地区名，我只需要保存其一次就可以了。我们举例说明，为了表示方便，我们用小写字母代表 IP 记录，C 表示国家名，A 表示地区名：

1. 有两条记录 a(C1, A1), b(C2, A2)，如果 C1 = C2, A1 = A2，那么我们就可以使用图 3 显示的结构来实现重用
2. 有三条记录 a(C1, A1), b(C2, A2), c(C3, A3)，如果 C1 = C2, A2 = A3，现在我们想存储记录 b，那么我们可以用图 6 的结构来实现重用
3. 有两条记录 a(C1, A1), b(C2, A2)，如果 C1 = C2，现在我们想存储记录 b，那么我们可以采用模式 2 表示 C2，用字符串表示 A2

你可以举出更多的情况，你也会发现在这种结构下，不同的字符串只需要存储一次。

了解索引区

在"了解文件头"部分，我们说明了文件头实际上是两个指针，分别指向了第一条索引和最后一条索引的绝对偏移。如图 8 所示：

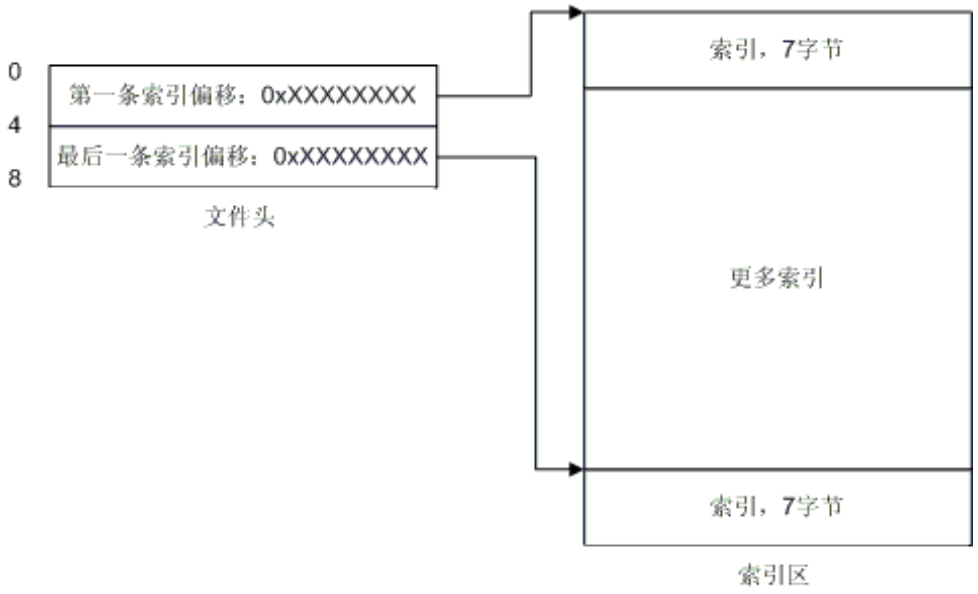

图 8. 文件头指向索引区图示

实在是很简单，不是吗？从文件头你就可以定位到索引区，然后你就可以开始搜索 IP 了！每条索引长度为 7 个字节，前 4 个字节是起始 IP 地址，后三个字节就指向了 IP 记录。这里有些概念需要说明一下，什么是起始 IP，那么有没有结束 IP？ 假设有这么一条记录：166.111.0.0 - 166.111.255.255，那么 166.111.0.0 就是起始 IP，166.111.255.255 就是结束 IP，结束 IP 就是 IP 记录中的那头 4 个字节，这下你应该就清楚了吧。于是乎，每条索引配合一条记录，构成了一个 IP 范围，如果你要查找 166.111.138.138 所在的位置，你就会发现 166.111.138.138 落在了 166.111.0.0 - 166.111.255.255 这个范围内，那么你就可以顺着这条索引去读取国家和地区名了。那么我们给出一个最详细的图解吧：

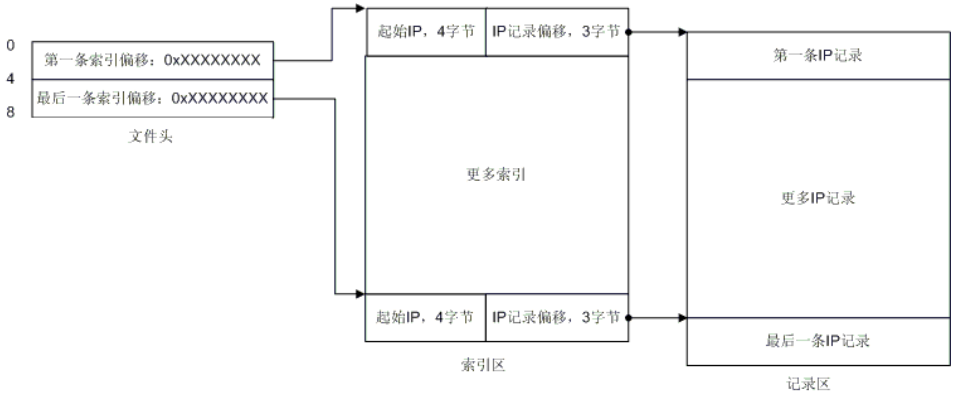

图 9. 文件详细结构

现在一切都清楚了是不是？也许还有一点你不清楚，QQWry.dat 的版本信息存在哪里呢？ 答案是：最后一条 IP 记录实际上就是版本信息，最后一条记录显示出来就是这样：255.255.255.0 255.255.255.255 纯真网络 2004 年 6 月 25 日 IP 数据。OK，到现在你应该全部清楚了。

Demo

下一步:我给出一个读取 IP 记录的程序片断,此片断摘录自 LumaQQ 源文件 edu.tsinghua.lumaqq.IPSeeker.java，如果你有兴趣，可以下载源代码详细看看。

```
/**
 * 给定一个 ip 国家地区记录的偏移，返回一个 IPLocation 结构
 * @param offset 国家记录的起始偏移
 * @return IPLocation 对象
 */
private IPLocation getIPLocation(long offset) {
 try {
  // 跳过 4 字节 ip
  ipFile.seek(offset + 4);
  // 读取第一个字节判断是否标志字节
  byte b = ipFile.readByte();
  if(b == REDIRECT_MODE_1) {
   // 读取国家偏移
   long countryOffset = readLong3();
   // 跳转至偏移处
   ipFile.seek(countryOffset);
   // 再检查一次标志字节，因为这个时候这个地方仍然可能是个重定向
   b = ipFile.readByte();
   if(b == REDIRECT_MODE_2) {
    loc.country = readString(readLong3());
    ipFile.seek(countryOffset + 4);
   } else
    loc.country = readString(countryOffset);
   // 读取地区标志
   loc.area = readArea(ipFile.getFilePointer());
  } else if(b == REDIRECT_MODE_2) {
```

```
    loc.country = readString(readLong3());
    loc.area = readArea(offset + 8);
   } else {
    loc.country = readString(ipFile.getFilePointer() - 1);
    loc.area = readArea(ipFile.getFilePointer());
   }
   return loc;
  } catch (IOException e) {
   return null;
  }
 }

 /**
  * 从 offset 偏移开始解析后面的字节，读出一个地区名
  * @param offset 地区记录的起始偏移
  * @return 地区名字符串
  * @throws IOException 地区名字符串
  */
 private String readArea(long offset) throws IOException {
  ipFile.seek(offset);
  byte b = ipFile.readByte();
  if(b == REDIRECT_MODE_1 || b == REDIRECT_MODE_2) {
   long areaOffset = readLong3(offset + 1);
   if(areaOffset == 0)
    return LumaQQ.getString("unknown.area");
   else
    return readString(areaOffset);
  } else
   return readString(offset);
 }
```

```
/**
 * 从 offset 位置读取 3 个字节为一个 long，因为 java 为 big-endian 格式，所以没办法
 * 用了这么一个函数来做转换
 * @param offset 整数的起始偏移
 * @return 读取的 long 值，返回-1 表示读取文件失败
 */
private long readLong3(long offset) {
 long ret = 0;
 try {
  ipFile.seek(offset);
  ipFile.readFully(b3);
  ret |= (b3[0] & 0xFF);
  ret |= ((b3[1] << 8) & 0xFF00);
  ret |= ((b3[2] << 16) & 0xFF0000);
  return ret;
 } catch (IOException e) {
  return -1;
 }
}

/**
 * 从当前位置读取 3 个字节转换成 long
 * @return 读取的 long 值，返回-1 表示读取文件失败
 */
private long readLong3() {
 long ret = 0;
 try {
  ipFile.readFully(b3);
  ret |= (b3[0] & 0xFF);
```

```
    ret |= ((b3[1] << 8) & 0xFF00);
    ret |= ((b3[2] << 16) & 0xFF0000);
    return ret;
   } catch (IOException e) {
    return -1;
   }
  }

  /**
   * 从 offset 偏移处读取一个以 0 结束的字符串
   * @param offset 字符串起始偏移
   * @return 读取的字符串，出错返回空字符串
   */
  private String readString(long offset) {
   try {
    ipFile.seek(offset);
    int i;
    for(i = 0, buf[i] = ipFile.readByte(); buf[i] != 0; buf[++i] = ipFile.readByte());
    if(i != 0)
        return Utils.getString(buf, 0, i, "GBK");
   } catch (IOException e) {
       log.error(e.getMessage());
   }
   return "";
  }
```

代码并不复杂，getIPLocation 是主要方法，它检查国家记录格式，并针对字符串形式，模式 1，模式 2 采用不同的代码，readArea 则相对简单，因为只有字符串和重定向两种情况需要处理。

总结

纯真 IP 数据库的结构使得查找 IP 简单迅速，不过你想要编辑它却是比较麻烦的，我想应该需要专门的工具来生成 QQWry.dat 文件，由于其文件格式的限制，你要直接添加 IP 记录就不容易了。不过，能查到 IP 已经很开心了，希望纯真记录越来越多～。