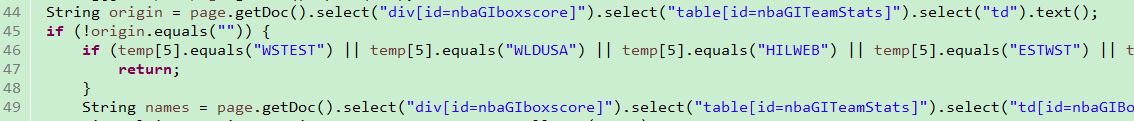
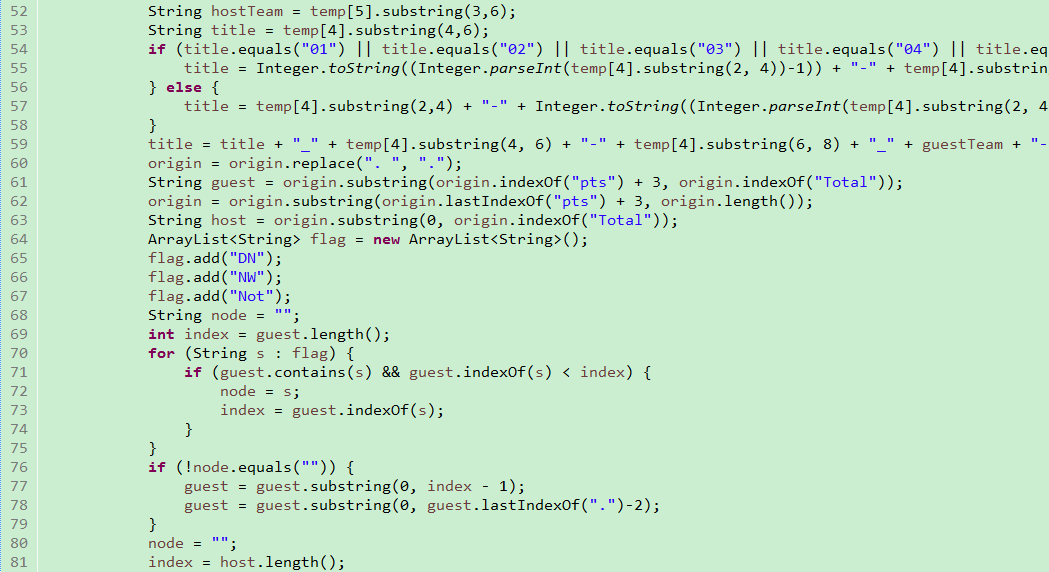
数据说明

南京大学Daisy小组

我们爬取[www.nba.com](http://www.nba.com)网站的数据，采用webcollector爬虫进行网页广度遍历爬取，首先设定爬取开始和结束的时间，爬虫自动添加seed，然后找到单场比赛的网页，找到网页后自动调用visit方法进行网页的处理，首先得到整个网页的DOM树，然后通过Tag区分，得到所需要的数据，比如我们得到单场比赛球员技术统计，这些数据被保存在表格里，我们先通过div[id=nbaGIboxscore]找到表格所在的部分，然后再在其中通过table[id=nbaGITeamStats]筛取我们需要的表格，最后筛取表格的每一行并保存下来，完成了对一个网页的操作。



我们爬取了单场比赛的球员技术统计，由于官网上球员名是简写的，比如LeBron James在官网上显示为L. James，我们原来想的是再爬一层，即爬取球员个人主页把球员名字取出，但由于我们采用的是广度遍历爬虫，在爬取晚所有比赛数据之后才能爬取球员个人信息页，考虑到这样要保存的数据量以及效率的问题，我们决定直接找到源代码里面指示球员个人信息页的链接并适当变换来获得球员完整名称，处理每场比赛时，先找到球员名，把简写和完整的名字保存在一个map中，到时候根据简写就能得到完整名字。我们还发现官网有些球员的名字没有链接，就是官网上已经查不到这个球员的信息，对于这种情况，我们直接保存了球员的简写，当然这种情况并不多见。另外，由于通过上面的初步筛选，还有些特殊情况，比如总分，没上场的球员等要处理，我们通过一些flag来处理，首先在得到的原始数据中通过split筛去总分，在通过未上场球员的情况如DNP,NWT等标志筛去未上场的球员，这样就得到了较为完善的球员数据，最后，适当调换一下位置，并选取所需的数据，完成对一个网页的操作。处理完数据后，我们按照原来迭代一迭代二的格式把爬取到的数据保存起来以便数据库初始化。



然后关于数据库的设计，我们的想法是把数据直接存入数据库，并按照赛季，常规赛季后赛区分数据表，这样设计有利于比赛和赛季的查询。其中在matches数据表中存比赛的日期，主客场，总比分以及各节比分。一场比赛产生两条数据，分别为主场球队数据和客场球队数据。在playerdata中存储一场比赛中每个球员的详细技术统计，这样便于计算每个球员的个人信息，其中，我们设置date,team,name为主键来确定唯一性。

另外，考虑到数据查询的效率，我们增加了teamsum和playersum来保存球队和球员一个赛季的总数据，如果要得到总数据，直接在这些表中查询即可。但这带来的不利因素就是数据库初始化，即把数据从文件读入数据库时比较费时。针对这个问题，我们采用批量插入的方法，尽可能提高效率。不过这只会在数据初始化的时候遇到，在初始化完就不会有这个问题了。

