Phase 2 Report: Web data scraping

Shanshan Zhang, tuf14438@temple.edu

I. Data Schema

There are the following 6 tables in the PubWorld database. Detailed description is in Phase 1 Report. Papers table is added with one new column named Year, which is for the sake of efficient queries, e.g. with this new column, it will be more efficient for the query 'find all short papers from 2011'. Paper_Author is added with one new column name AuthorPos, which is the positions of corresponding authors in one paper. Conference is added with a new column named Location. Column names in bold are primary key of the table.

- Authors: **AuthorID**, FirstName, LastName, FullName, Affiliation.
- Papers: **PaperID**, Title (name of paper), PaperNo (paper number in the conference), Track (IR, DB, KM), Type (Regular, Full, Short, Demo) , Year
- Paper_Author: **ID**, PaperID, AuthorID, ConfName, Year, AuthorPos.
- Conference: **ConfName, Year**, Location.
- PCMembers: **PCMemberID**, FirstName, LastName, FullName, Affiliation.

The fields highlighted are created based on the feedback from the Phase 1 and from new knowledge gained in Phase 2.

II. Data Extraction Algorithm

1. Problem Definition

When looking at CIKM 2012 - 2013, at least 9 web pages should be crawled, they are:

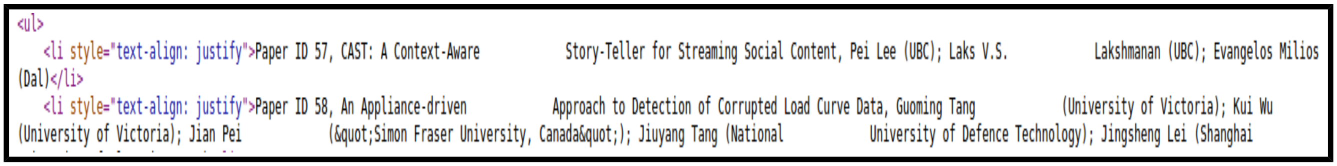
- 2 pages for 2012: program committee page, and accepted papers (including all types of papers) page.
- 3 pages for 2013: program committee page, accepted papers (including only regular papers), accepted poster papers, accepted demo papers.
- 4 pages for 2014: program committee page, accepted papers (including only regular and demo papers), and accepted poster papers.

The difference in formatting exists for pages from different years. For example, in year 2013 and year 2014, paper records or committee records are stored with <ul> and <li> tags, while in year 2012, the records are organized with <table><tr><td> tags. Figure 1 and 3 demonstrates two sample paper records of these two types of organization.

The way in Figure 1 creates a need of wisely tokening a string if we want to separate Authors table from Papers table, while the second way in Figure 2 is so self-organized that we can easily get Authors table and Papers table with only a little effort.

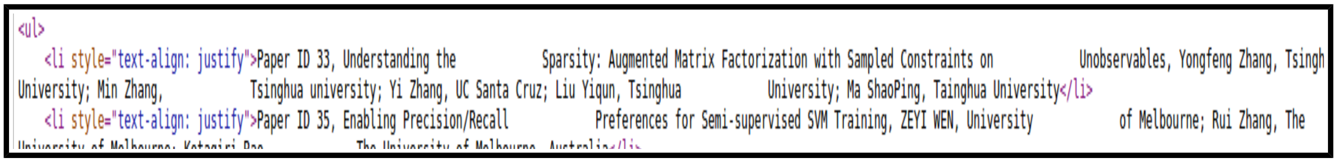
The formatting difference can also exist within the same year as illustrated in Figure 1 and Figure 3. They both come from the accepted papers page in 2014.

In this project, there should be two tasks: a). Extraction contents from the website body. b). Insert contents into corresponding tables in the database.



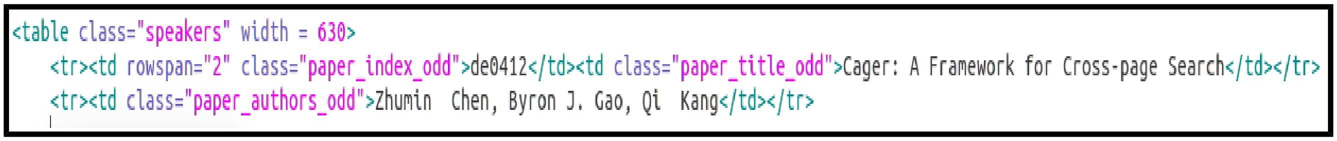
```
<ul>
   <li style="text-align: justify">Paper ID 57, CAST: A Context-Aware          Story-Teller for Streaming Social Content, Pei Lee (UBC); Laks V.S.          Lakshmanan (UBC); Evangelos Milios
(Dal)</li>
   <li style="text-align: justify">Paper ID 58, An Appliance-driven          Approach to Detection of Corrupted Load Curve Data, Guoming Tang          (University of Victoria); Kui Wu
(University of Victoria); Jian Pei          (&quot;Simon Fraser University, Canada&quot;); Jiuyang Tang (National          University of Defence Technology); Jingsheng Lei (Shanghai
```

Figure 1



```
<ul>
    <li style="text-align: justify">Paper ID 33, Understanding the           Sparsity: Augmented Matrix Factorization with Sampled Constraints on           Unobservables, Yongfeng Zhang, Tsingh
University; Min Zhang,           Tsinghua university; Yi Zhang, UC Santa Cruz; Liu Yiqun, Tsinghua           University; Ma ShaoPing, Tainghua University</li>
    <li style="text-align: justify">Paper ID 35, Enabling Precision/Recall           Preferences for Semi-supervised SVM Training, ZEYI WEN, University           of Melbourne; Rui Zhang, The
University of Melbourne; Kotagiri Rao           The University of Melbourne, Australia</li>
```

Figure 2



```
<table class="speakers" width = 630>
    <tr><td rowspan="2" class="paper_index_odd">de0412</td><td class="paper_title_odd">Cager: A Framework for Cross-page Search</td></tr>
    <tr><td class="paper_authors_odd">Zhumin  Chen, Byron J. Gao, Qi  Kang</td></tr>
```

Figure 3

2. Algorithm for extraction content

The various formats used in the 9 web pages are trivial but make it's impossible to use the same algorithm for all 9 pages. Thus we design 9 extraction algorithms, one for each page, with small or little change among them. In general, each algorithm will output a set of objects that can be further processed and inserted to MySQL tables. And each algorithm is composed by two functions: a) HTML tag anchoring function; b) string tokenization function. The record in Figure 2 will be taken as an example for explanation.

a) HTML tag anchoring function: direct the file reader pointer to the start position of the desired lines in the HTML. For example, the anchoring function will process all records line by line between <li> and </li> element, which are children of an <ul> element in Figure 2.

b) String tokenization function: Each record from a) will be passed to this function as a string. The string is split by some punctuations first and can be split or concatenated several times if needed. For the string example: . It's first split by the comma “,”. Then the following list of tokens *[{Paper ID 33}, {Understanding the Sparsity: Augmented Matrix Factorization with Sampled Constraints on Unobservables}, {Yongfeng Zhang} , {Tsinghua University; Min Zhang} , {Tsinghua university; Yi Zhang} , {UC Santa Cruz; Liu Yiqun}, {Tsinghua University; Ma ShaoPing}, {Tainghua University}]* are obtained. The first token {Paper ID 33} and second tokens {Understanding the Sparsity: Augmented Matrix Factorization with Sampled Constraints on Unobservables} will be directly stored in an object. All tokens from the third on to the last one of the list will be concatenated by comma “,” first and split by semicolon “;” again. Then we get a new list containing all authors [*{Yongfeng Zhang, Tsinghua University}, {Min Zhang, Tsinghua university}, {Yi Zhang, UC Santa Cruz}, {Liu Yiqun, Tsinghua University}, {Ma ShaoPing, Tainghua University}*]. Each author in the list will be split and concatenate again until we get their corresponding first name, last name, affiliation. All these info will be stored in the object. This object will finally have several variables like paper number, paper name, authors list, etc.

3. Algorithm for inserting MySQL

Once we had the output objects, connected to the database, we need to insert the objects into the corresponding tables in MySQL. Two conditions should be fulfilled when inserting into

a). Before inserting a row into a table, existence of the row must be checked. For example, if there is already an author named 'Min Zhang' and from 'Tsinghua university', we should get the existing AuthorID. The granularity should be as detailed as possible in order not to loss information.

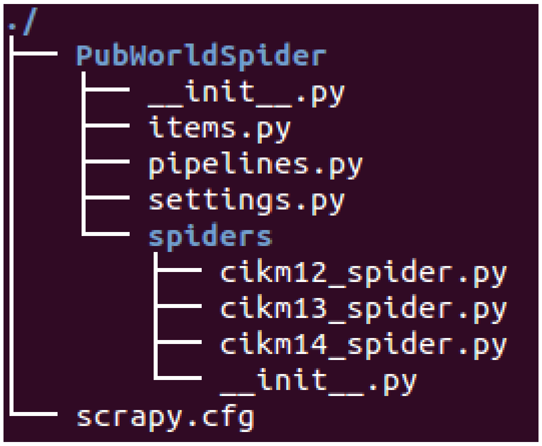
b). Papers and Authors tables should be inserted before Paper_Author is inserted since Paper_Author contain foreign keys referring to the other two tables. Similarly, Conference and PCMembers tables should be inserted before PC_Conf table.

III. Implementation using Scrapy framework

I used the **Scrapy** framework for implementation. Scrapy is a web crawler framework in Python. Introduction and demos can be found at *http://doc.scrapy.org/en/latest/intro/tutorial.html*

1. Project hierarchy and explanation

Table 1 shows the project folder hierarchy. Spider folder contains 9 spiders, i.e. the 9 content extraction algorithms for the 9 web pages and they are written into 3 files correspondingly. *pipelines.py* file contains algorithm for inserting into tables that inserts the variables of items, i.e. objects, into the corresponding columns of tables in MySQL. The *items.py* file contains all item definitions. And the *setting.py* configures which item should use which inserting algorithm.

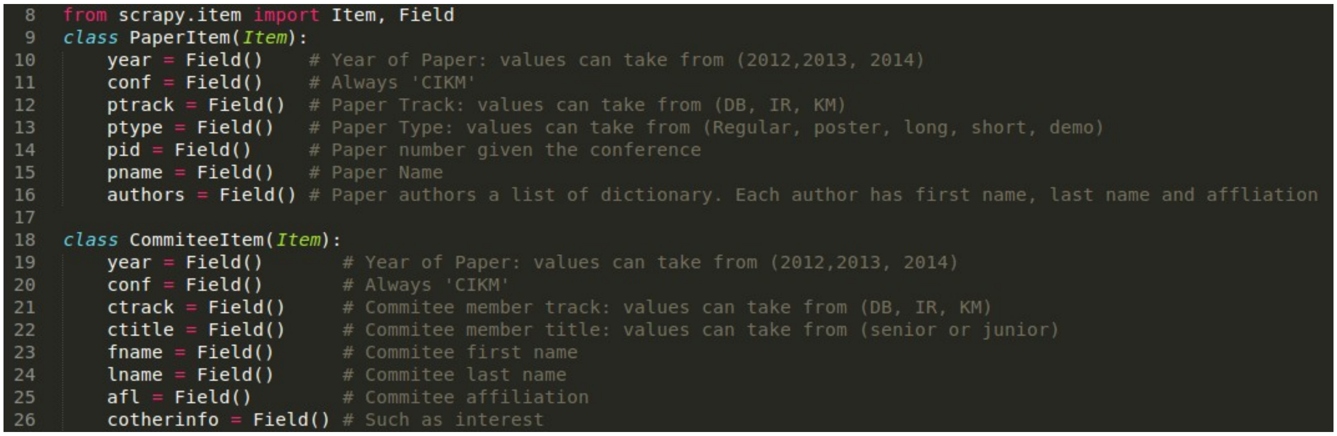


```
./
├── PubWorldSpider
│   ├── __init__.py
│   ├── items.py
│   ├── pipelines.py
│   ├── settings.py
│   └── spiders
│       ├── cikm12_spider.py
│       ├── cikm13_spider.py
│       ├── cikm14_spider.py
│       └── __init__.py
└── scrapy.cfg
```

Figure 4

2. Design of Items

There are two items defined as shown in Figure 5. And all details are also written in the comments part in the Figure 5.



```python
from scrapy.item import Item, Field
class PaperItem(Item):
    year = Field()    # Year of Paper: values can take from (2012,2013, 2014)
    conf = Field()    # Always 'CIKM'
    ptrack = Field()  # Paper Track: values can take from (DB, IR, KM)
    ptype = Field()   # Paper Type: values can take from (Regular, poster, long, short, demo)
    pid = Field()     # Paper number given the conference
    pname = Field()   # Paper Name
    authors = Field() # Paper authors a list of dictionary. Each author has first name, last name and affliation

class CommiteeItem(Item):
    year = Field()       # Year of Paper: values can take from (2012,2013, 2014)
    conf = Field()       # Always 'CIKM'
    ctrack = Field()     # Commitee member track: values can take from (DB, IR, KM)
    ctitle = Field()     # Commitee member title: values can take from (senior or junior)
    fname = Field()      # Commitee first name
    lname = Field()      # Commitee last name
    afl = Field()        # Commitee affiliation
    cotherinfo = Field() # Such as interest
```

IV. Accuracy

The accuracy for 2012 is 100 %, because all columns I needed for the tables are highly formatted using HTML <td> tags. For 2013 and 2014 data, the accuracy is at least 99.9999% as far as I concerned. Because Scrapy itself can locate the records with 100% success, and I cleaned all punctuations that may cause parsing problems before I parse each record.

V. Results and statistics

The cardinality for each table is shown in Table 1. And some statistics are shown in Table 2.

Table 1. Cardinality of each table

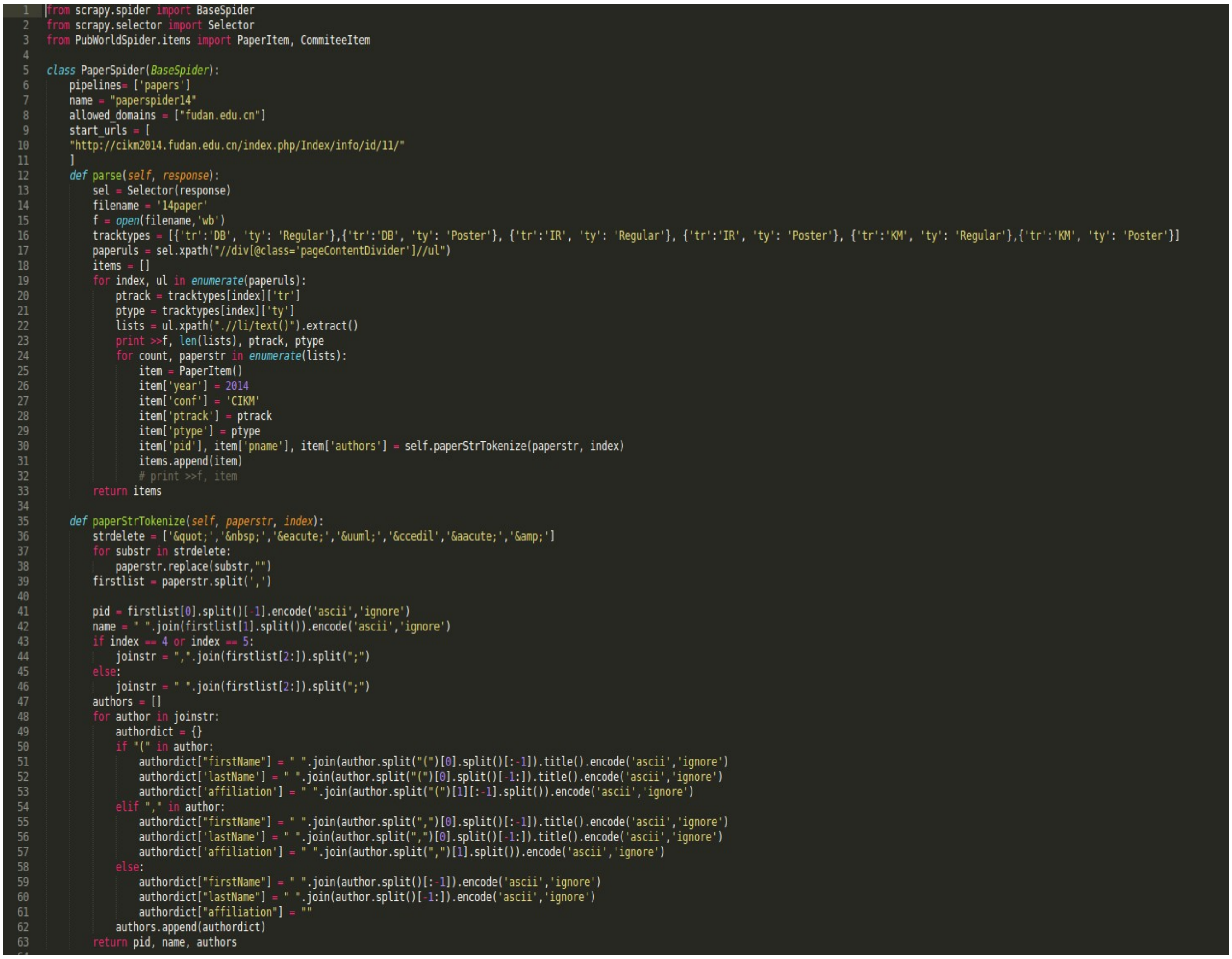
	Papers	Authors	Paper_Author	PCMembers	Conference	PC_Conf
Cardinality	1049	3042	3670	1360	3	1427

Table 2. Statistics for each conference year

	Number of Papers	Number of Authors	Number of PCMembers
2012	435	1326	442
2013	351	1049	505
2014	263	828	456

Appendix. Sample codes

A. Sample Code of extraction content algorithm.



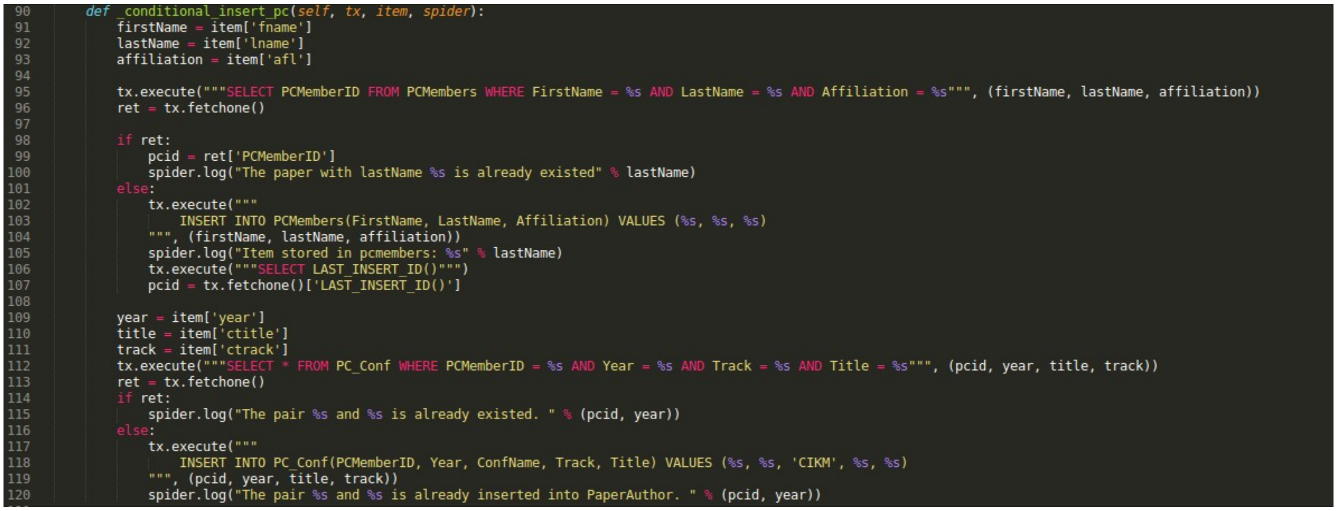
```python
from scrapy.spider import BaseSpider
from scrapy.selector import Selector
from PubWorldSpider.items import PaperItem, CommiteeItem

class PaperSpider(BaseSpider):
    pipelines= ['papers']
    name = "paperspider14"
    allowed_domains = ["fudan.edu.cn"]
    start_urls = [
    "http://cikm2014.fudan.edu.cn/index.php/Index/info/id/11/"
    ]
    def parse(self, response):
        sel = Selector(response)
        filename = '14paper'
        f = open(filename,'wb')
        tracktypes = [{'tr':'DB', 'ty': 'Regular'},{'tr':'DB', 'ty': 'Poster'}, {'tr':'IR', 'ty': 'Regular'}, {'tr':'IR', 'ty': 'Poster'}, {'tr':'KM', 'ty': 'Regular'},{'tr':'KM', 'ty': 'Poster'}]
        paperuls = sel.xpath("//div[@class='pageContentDivider']//ul")
        items = []
        for index, ul in enumerate(paperuls):
            ptrack = tracktypes[index]['tr']
            ptype = tracktypes[index]['ty']
            lists = ul.xpath(".//li/text()").extract()
            print >>f, len(lists), ptrack, ptype
            for count, paperstr in enumerate(lists):
                item = PaperItem()
                item['year'] = 2014
                item['conf'] = 'CIKM'
                item['ptrack'] = ptrack
                item['ptype'] = ptype
                item['pid'], item['pname'], item['authors'] = self.paperStrTokenize(paperstr, index)
                items.append(item)
                # print >>f, item
        return items

    def paperStrTokenize(self, paperstr, index):
        strdelete = ['&quot;','&nbsp;','&eacute;','&uuml;','&ccedil','&aacute;','&amp;']
        for substr in strdelete:
            paperstr.replace(substr,"")
        firstlist = paperstr.split(',')

        pid = firstlist[0].split()[-1].encode('ascii','ignore')
        name = " ".join(firstlist[1].split()).encode('ascii','ignore')
        if index == 4 or index == 5:
            joinstr = ",".join(firstlist[2:]).split(";")
        else:
            joinstr = " ".join(firstlist[2:]).split(";")
        authors = []
        for author in joinstr:
            authordict = {}
            if "(" in author:
                authordict["firstName"] = " ".join(author.split("(")[0].split()[:-1]).title().encode('ascii','ignore')
                authordict['lastName'] = " ".join(author.split("(")[0].split()[-1:]).title().encode('ascii','ignore')
                authordict['affiliation'] = " ".join(author.split("(")[1][:-1].split()).encode('ascii','ignore')
            elif "," in author:
                authordict["firstName"] = " ".join(author.split(",")[0].split()[:-1]).title().encode('ascii','ignore')
                authordict['lastName'] = " ".join(author.split(",")[0].split()[-1:]).title().encode('ascii','ignore')
                authordict['affiliation'] = " ".join(author.split(",")[1].split()).encode('ascii','ignore')
            else:
                authordict["firstName"] = " ".join(author.split()[:-1]).encode('ascii','ignore')
                authordict["lastName"] = " ".join(author.split()[-1:]).encode('ascii','ignore')
                authordict["affiliation"] = ""
            authors.append(authordict)
        return pid, name, authors
```

B. Sample Code of inserting MySQL.



```python
    def _conditional_insert_pc(self, tx, item, spider):
        firstName = item['fname']
        lastName = item['lname']
        affiliation = item['afl']

        tx.execute("""SELECT PCMemberID FROM PCMembers WHERE FirstName = %s AND LastName = %s AND Affiliation = %s""", (firstName, lastName, affiliation))
        ret = tx.fetchone()

        if ret:
            pcid = ret['PCMemberID']
            spider.log("The paper with lastName %s is already existed" % lastName)
        else:
            tx.execute("""
                INSERT INTO PCMembers(FirstName, LastName, Affiliation) VALUES (%s, %s, %s)
            """, (firstName, lastName, affiliation))
            spider.log("Item stored in pcmembers: %s" % lastName)
            tx.execute("""SELECT LAST_INSERT_ID()""")
            pcid = tx.fetchone()['LAST_INSERT_ID()']

        year = item['year']
        title = item['ctitle']
        track = item['ctrack']
        tx.execute("""SELECT * FROM PC_Conf WHERE PCMemberID = %s AND Year = %s AND Track = %s AND Title = %s""", (pcid, year, title, track))
        ret = tx.fetchone()
        if ret:
            spider.log("The pair %s and %s is already existed. " % (pcid, year))
        else:
            tx.execute("""
                INSERT INTO PC_Conf(PCMemberID, Year, ConfName, Track, Title) VALUES (%s, %s, 'CIKM', %s, %s)
            """, (pcid, year, title, track))
            spider.log("The pair %s and %s is already inserted into PaperAuthor. " % (pcid, year))
```

Phase 1 Report: DBMS Installation and Working Dataset

Shanshan Zhang, tuf14438@temple.edu

I. Data Description

The conference chosen for analysis is CIKM in 3 consecutive years.

In the preliminary analysis phase, there should be at least individual tables for the following entities: **Papers, Authors, PCMembers** for program committee members. A paper has columns like title, author, year, track (*IR, KM or DB*), topic, paper number, type (*full, long, demo, poster*). An author has columns like first name, last name, affiliation. A committee member has the same columns with an author. While when analyzing in depth, there are following relationships exist among these entities:

- Many-to-many relationship: 1) one paper is co-authored by multiple authors and one author can write multiple papers. 2) one PC member may appear in multiple conferences and one conferences have multiple PC members.

To address the two many-to-many relationship, I added a table called **Paper_Author** which stores only the paper-author pairs and using the PaperID and AuthorID as the foreign keys. And another table called **PC_Conf** table which stores the PCMember-conference pairs and using **PCMemberID** as the foreign key.

There then comes another question: where to put the year/conference information. For papers, one paper can appear in only one year, while for a committee member, he/she can appear in several years.

I considered three ways to add the year/conference information:

1. Add a column called year to **Papers** table and **PC_Conf** table.

2. Add a column called year to **Paper_Author** table and **PC_Conf** table.

3. Add a table called **Conferences** with two columns conferene name and year. The primary key contains the two columns and the two columns are also added to **Paper_Author** and **PC_Conf** tables as froreign keys.

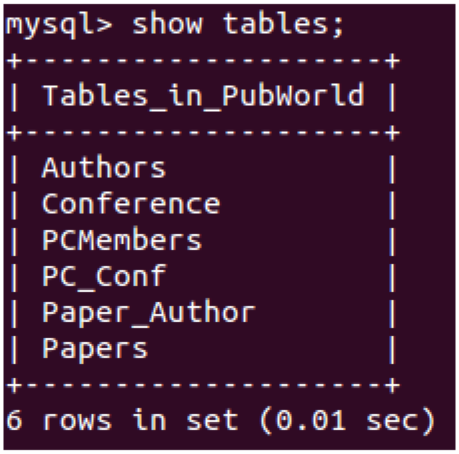
Either the three is enough for my application, because I chose the same conference in 3 consecutive years. While finally I chose strategy 3 for the following reasons:

1. More efficiency for some query task. For example, if I want to analyze how many papers are published by every author in a specific year, I can only refer to the **Paper_Author** table without any other merging or joining operations.

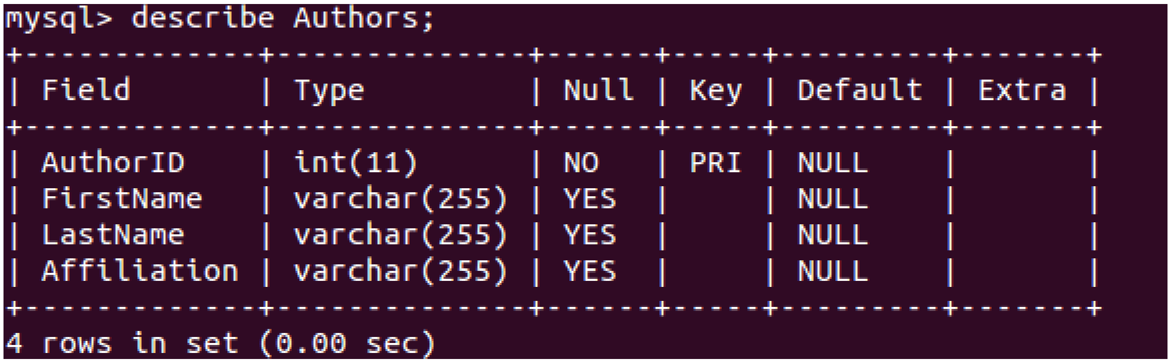
2. Scalability. If later on I need to parse more conferences, strategy 3 will be more scalable since what I need to do is adding more rows in every table without altering the structure of tables.

One last minor consideration is whether there is a need to separate **Authors** and **PCMembers** table because they have exactly the same columns. So far, I didn't see any hurt of separating them, so I will keep them separated now.

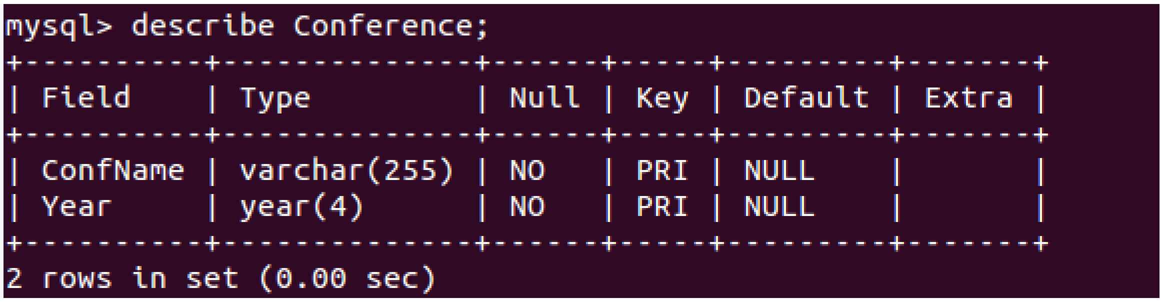
II. Tables



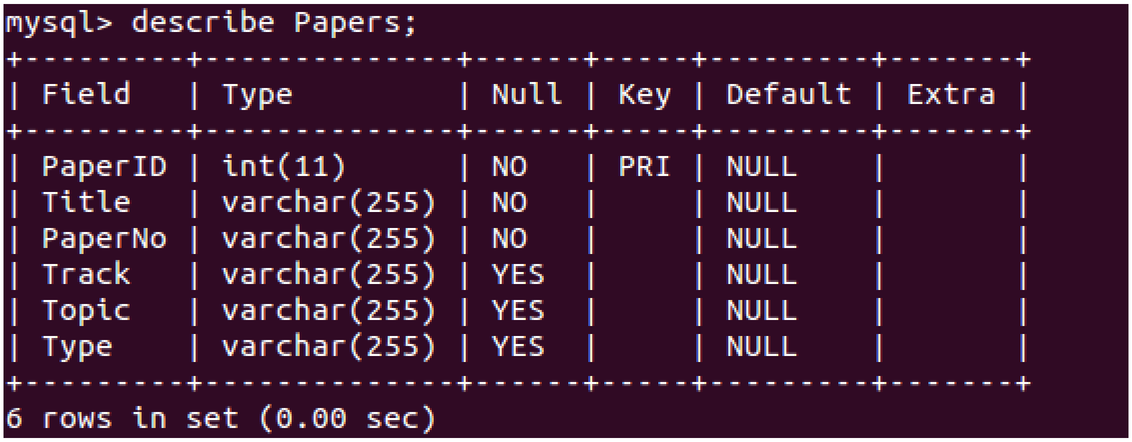
```
mysql> show tables;
+---------------------+
| Tables_in_PubWorld |
+---------------------+
| Authors             |
| Conference          |
| PCMembers           |
| PC_Conf             |
| Paper_Author        |
| Papers              |
+---------------------+
6 rows in set (0.01 sec)
```



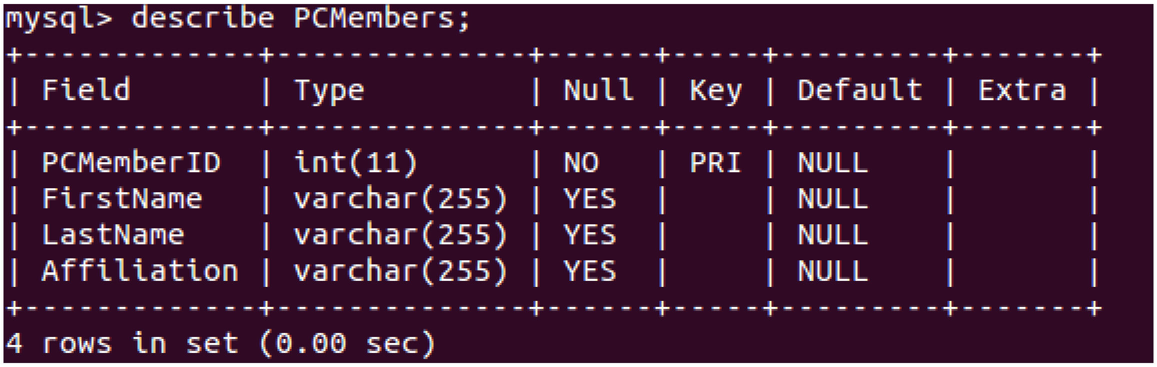
```
mysql> describe Authors;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| AuthorID    | int(11)      | NO   | PRI | NULL    |       |
| FirstName   | varchar(255) | YES  |     | NULL    |       |
| LastName    | varchar(255) | YES  |     | NULL    |       |
| Affiliation | varchar(255) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```



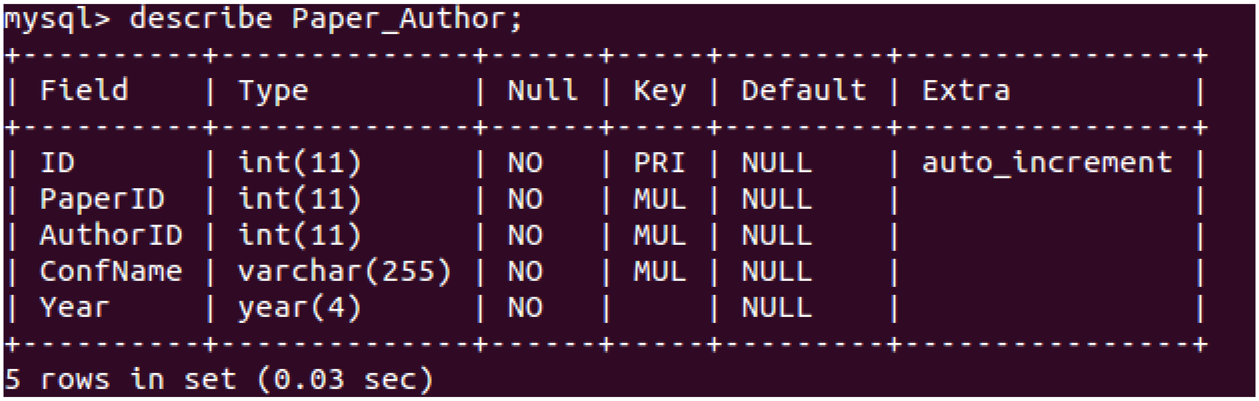
```
mysql> describe Conference;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| ConfName | varchar(255) | NO   | PRI | NULL    |       |
| Year     | year(4)      | NO   | PRI | NULL    |       |
+----------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```



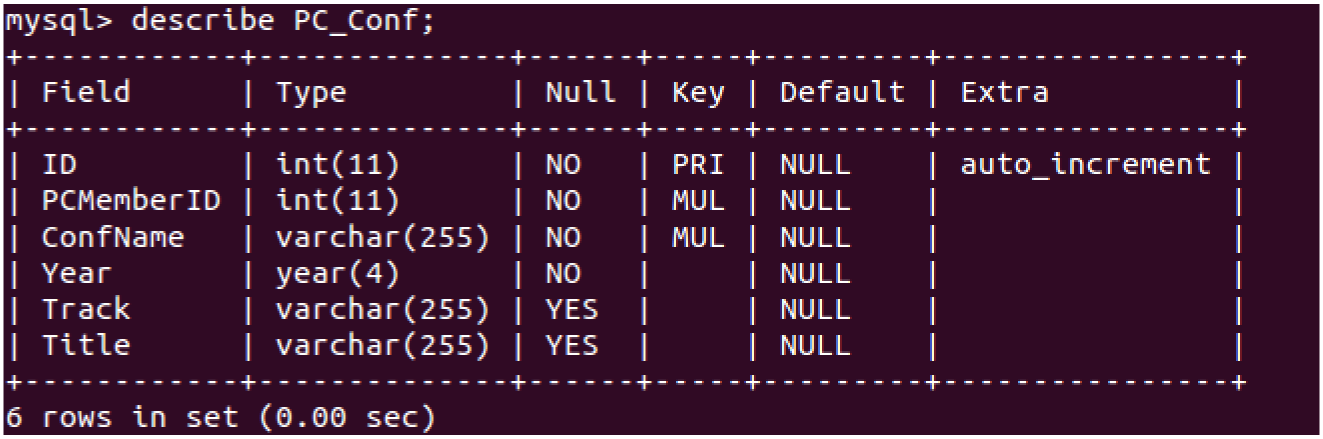
```
mysql> describe Papers;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| PaperID | int(11)      | NO   | PRI | NULL    |       |
| Title   | varchar(255) | NO   |     | NULL    |       |
| PaperNo | varchar(255) | NO   |     | NULL    |       |
| Track   | varchar(255) | YES  |     | NULL    |       |
| Topic   | varchar(255) | YES  |     | NULL    |       |
| Type    | varchar(255) | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```



```
mysql> describe PCMembers;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| PCMemberID  | int(11)      | NO   | PRI | NULL    |       |
| FirstName   | varchar(255) | YES  |     | NULL    |       |
| LastName    | varchar(255) | YES  |     | NULL    |       |
| Affiliation | varchar(255) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```



```
mysql> describe Paper_Author;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| ID       | int(11)      | NO   | PRI | NULL    | auto_increment |
| PaperID  | int(11)      | NO   | MUL | NULL    |                |
| AuthorID | int(11)      | NO   | MUL | NULL    |                |
| ConfName | varchar(255) | NO   | MUL | NULL    |                |
| Year     | year(4)      | NO   |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
5 rows in set (0.03 sec)
```



```
mysql> describe PC_Conf;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| ID         | int(11)      | NO   | PRI | NULL    | auto_increment |
| PCMemberID | int(11)      | NO   | MUL | NULL    |                |
| ConfName   | varchar(255) | NO   | MUL | NULL    |                |
| Year       | year(4)      | NO   |     | NULL    |                |
| Track      | varchar(255) | YES  |     | NULL    |                |
| Title      | varchar(255) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

III. Scripts

```
##################################################
# @Author: Shanshan Zhang
# @Date: 09/16/2014
# @Class: Principle of Data Management
# @Title: MySQL script for Phase 1.
##################################################

CREATE DATABASE PubWorld;
USE PubWorld;
SHOW TABLES;

CREATE TABLE IF NOT EXISTS Conference
(
ConfName VARCHAR(255) NOT NULL,
Year YEAR(4) NOT NULL,
PRIMARY KEY (ConfName, Year)
) ENGINE=INNODB;

CREATE TABLE IF NOT EXISTS Papers
(
PaperID INT NOT NULL,
Title VARCHAR(255) NOT NULL,
PaperNo VARCHAR(255) NOT NULL,
Track VARCHAR(255),
Topic VARCHAR(255),
Type VARCHAR(255),
PRIMARY KEY (PaperID)
) ENGINE=INNODB;

CREATE TABLE IF NOT EXISTS Authors
(
AuthorID INT NOT NULL,
FirstName VARCHAR(255),
LastName VARCHAR(255),
Affiliation VARCHAR(255),
PRIMARY KEY (AuthorID)
) ENGINE=INNODB;

CREATE TABLE IF NOT EXISTS PCMembers
(
PCMemberID INT NOT NULL,
FirstName VARCHAR(255),
LastName VARCHAR(255),
Affiliation VARCHAR(255),
PRIMARY KEY(PCMemberID)
) ENGINE=INNODB;
CREATE TABLE IF NOT EXISTS Paper_Author
(
ID INT NOT NULL AUTO_INCREMENT,
PaperID INT NOT NULL,
AuthorID INT NOT NULL,
ConfName VARCHAR(255) NOT NULL,
Year YEAR(4) NOT NULL,
PRIMARY KEY (ID),
INDEX (PaperID),
INDEX (AuthorID),
INDEX (ConfName, Year),
FOREIGN KEY (PaperID)
      REFERENCES Papers (PaperID)
      ON UPDATE CASCADE ON DELETE RESTRICT,
FOREIGN KEY (AuthorID)
      REFERENCES Authors (AuthorID)
      ON UPDATE CASCADE ON DELETE RESTRICT,
FOREIGN KEY (ConfName, Year)
      REFERENCES Conference (ConfName, Year)
      ON UPDATE CASCADE ON DELETE RESTRICT
) ENGINE=INNODB;

--
CREATE TABLE IF NOT EXISTS PC_Conf
(
ID INT NOT NULL AUTO_INCREMENT,
PCMemberID INT NOT NULL,
ConfName VARCHAR(255) NOT NULL,
Year YEAR(4) NOT NULL,
Track VARCHAR(255),
Title VARCHAR(255),
PRIMARY KEY(ID),
INDEX (PCMemberID),
INDEX (ConfName, Year),
FOREIGN KEY(PCMemberID)
      REFERENCES PCMembers(PCMemberID)
      ON UPDATE CASCADE ON DELETE RESTRICT,
FOREIGN KEY(ConfName, Year)
      REFERENCES Conference(ConfName, Year)
      ON UPDATE CASCADE ON DELETE RESTRICT
) ENGINE=INNODB;
```