The Sloan Digital Sky Survey Datasets

Haoran Yu

1. DATA DESCRIPTION

Sloan Digital Sky Survey (SDSS) is an astronomy project that aims to provide a digital map of the sky [7]. SDSS data includes raw images of the sky and measurements known as *scientific attributes*, which are numerical estimates of the physical attributes of objects in the images [7]. The scientific attributes are stored in a database system called the the Catalog Archive Server (CAS) database. Access to the CAS data is provided through several data access interfaces, including the SkySever website, which provides a web-based interface for writing SQL queries. There are also SQL templates provided by SkyServer to assist users in constructing queries [4, 7].

SQL queries submitted via the access interfaces to the CAS are logged and provided in the SkyServer logs data [6, 4, 5]. These logs include information about the IP address that submitted the query, the web agent, the time stamp, the SQL query statement, and the version of the database that it queried [7].

Generally, there are 3 datasets associated with the SDSS project: [1] [2] [3] [4] [5] [6]

CAS database. The CAS dataset stores the scientific attributes of celestial objects identified in the captured photos, the corresponding photos can be retrieved from the Data Archive Server (DAS) at the Fermilab physical sciences laboratory [7] The schema for the original CAS database is shown in Figure 1[7]. SDSS releases new versions of data yearly to update and expand the astronomy data, each version is called a Data Release (DR). A recent data release (Data Release 12) has expanded the original CAS database schema to 115 tables, and offers 59 views, 231 functions and 21 procedures for queries [4]. In DR12, tables such as PhotoObjAll have over 350 table attributes and has as many as 790 million tuples [1]. Among the tables, PhotoObjAll, SpecObjAll, and Profile store the primary scientific attributes of the photos and spectrographs of the celestial objects.

SkyServer logs dataset. The SkyServer logs dataset aggregates each attempt to access the web interfaces and submission of SQL queries since 2001 and is publicly accessible [6, 4, 5]. [8] The schema for this dataset is shown in Figure 2 . Logs are harvested from the server every hour and stored in 7 tables [6]. The WebLog and SqlLog tables, which store each web page visit and each SQL query submitted, can be accessed online[9]. The web logs have recorded 630 million webpage views[5] and has logged over 330 million submitted SQL queries [2] from 2001 to 2011.

Normalized SkyServer logs. The third dataset is a subset of the SkyServer logs dataset storing data from 2001 to 2011, this dataset is normalized to reduce its storage size[4]. The normalization process involves moving common items such as IP addresses in WebLog and query statements in SqlLog to a separate table, and replacing the items in the original WebLog and SqlLog with foreign keys to reference the items in the separate tables[4]. The schema for this dataset is shown in Figure 3. These separate tables are called ancillary tables, such as ipDomain, session, Webagent, and SqlStatement. The modified dataset is publicly accessible and was used for SkyServer traffic analysis and SQL usage analysis [6, 4]. There are 14 tables in total, and all tables have less than 30 attributes [4]. The normalized logs were later expanded to include data until 2015 in [2] but is not

[1]The report should clear, concise, accurate, and easy to follow

[2]This section should very briefly introduce the data. I've rewritten this part to convey only the important information..

[3]Regarding references, don't name authors, only reference their paper. Also, please check grammar throughout please.

[4]Before you had just mentioned the scientific attributes. Remember, any new concept that you introduce should be defined, along with a reference to the paper that defined it originally.

[5]You need to put labels for the figures, then you refer to the labels in the tex files, rather than name the figures in the text. There is no point in introducing the figures here, since they were not described in detail, the reader doesn't gain anything from looking at the figures at this point.

[6]for the datasets below, we need statistics for the most recent release of the data. So, for the CAS database, you need to say which release the schema corresponds to and how many tables, attributes, etc it contains. Similarly for the rest of the datasets.

[7]For all these figures, you need to cite the original paper you took it from

[8]The have three papers on this, first 5 years, then two on the first 10 years. You need to cite all three here.

[9]All the tables are accessible, why this statement? the web interface allows simple select queries on the two tables, but you have access to all the data if you download it.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

publicly available, therefore we will use dataset until 2011 to perform further SQL usage analysis.

2. SESSIONS

Summary of procedure for data processing in 10 years paper-1: Client IP addresses and agent strings: First, agent strings in the logs are associated with Web agents. These web agents are then classified into 5 classes: anonymous, bot, administrative service, user program, or Web browser. This is stored in the agent table. Then, the domains of the IP addresses in the web hits are resolved, and categorized into 1047 domain organizations. The domain organizations are further categorized into: research institution, university, college, community college, k-12 institution, ISP, non-ISP, and government organization. (this info may not be available to us.)

According to [4], a *hit* "comes from a Web browser, or a program that the user has written to download data, and has a command requesting a file type that delivers information to the end user. " [10] For weblogs , a hit is a pageview, and each row in a weblog is a hit. A pageview is indicated in the normalized weblog table with a boolean flag variable. For SQL query logs, a hit is a query.

As defined by [4], a *session* is an ordered sequence of hits from a single IP address, such that the gaps between hits in the sequence is no longer than 30 minutes. If an additional hit occurs after the 30 minutes gap for that IP, a new session is declared.

2.1 Tracing a Session

To study the SQL usage pattern by a session, we will trace each session to reconstruct the original queries that were submitted. This can be achieved by joining multiple tables in the normalized logs dataset. For example, a session table defines individual sessions and the sessionLogs table stores all requests of all sessions with the rankInSession attribute that orders all requests of each session sequentially. sessionLogs table also stores the necessary pointers to the WebLog and SqlLog tables to retrieve the original query. Studying the results returned by each query assists our usage pattern analysis. However, since the SqlLog does not store the results returned by submitting the query - it only stores the number of rows returned - we will need submit the query to Skyserver to retrieve the query results.

We provide an example for a session, that is, a sequence of hits/queries submitted by a user. We traced the session, and obtained the hits/queries of the session in the table below.

The method used to trace a session is described in pseudo-SQL. Here, we trace a session with sessionID 100000.

```
--TRACING A SESSION--
--Query1: find the IP address of a session sessionIpID,
     and how many queries were made in that session
     sqlqueries
select sessionID, sqlqueries, sessionIpID
into TEMP_TABLE1
from session.csv
where sessionID = 100000;

--Query2: find the sequence of the queries made in a
     session where each query is indicated by a rank
     rankInSession, and get the time that the query was
     made theTime.
select sl.sessionID, sl.rankInSession, sl.theTime
into TEMP_TABLE2
from sessionlog.csv
join TEMP_TABLE1 t1
where sl.sessionID = t1.sessionID;

--Query3: confirm that this IP is valid
select ip.ipID, ip.isvalid
into TEMP_TABLE3
from ipAll.csv ip
join TEMP_TABLE1 t1
where ip.ipID = t1.sessionIpID;

--Query4: get the ID of the SQL query statement
     statementID for each query made in a session.
--Also get the number of rows returned by submitting a
     particular query.
--Get the access interface i.e. "casjobs" or "public"
     that the request was sent.
--NOTE theTime in TEMP_TABLE2 and in sqllog.csv are in
     different granularities, need to preprocess
     TEMP_TABLE2 first
select sl.clientIpID, sl.theTime, sl.statementID, sl.
     rows, sl.logID, sl.access
into TEMP_TABLE4
from sqllog.csv sl
join TEMP_TABLE2 t2
join TEMP_TABLE1 t1
where sl.clientIpID = t1.sessionIpID
and sl.theTime = t2.theTime

--Query 5: get the SQL query statement identified by the
      statementID
select ss.statementID, ss.statement
into TEMP_TABLE5
from sqlstatement.csv ss
join TEMP_TABLE4 t4
where t4.statementID = ss.statementID


--FINDING THE WEBAGENT OF THE QUERY--
--Query: get the class of the web agent string, BOT, or
     ADMIN, etc.
--NOTE: this query does not work! No weblog.csv row with
      the correct time corresponding to sqllog.csv can
     be found
select ls.logID, wl.agentStringID, was.agentID, wa.class
into TEMP_TABLE_AGENT
from LogSource.csv ls
join TEMP_TABLE4 t4
join weblog.csv wl
join weagentstring.csv was
join WebAgent.csv wa
where t4.logID = ls.logID
and ls.logID = wl.logID
and wl.agentStringID = was.agentID
and was.agentID = wa.agentID
```

After tracing a session with sessionID 100000, the following SQL query statements were retrieved from the logs.

```
--statementID 48063987
SELECT u.up_name as name,
   '<a target=INFO href=http://cas.sdss.org/astrodr6/en/
        tools/explore/obj.asp?id=' + cast(x.objId as
        varchar(20)) + '>'+ cast(x.objId as varchar(20))
        + '</a>' as objID, p.ra, p.dec,
   dbo.fPhotoTypeN(p.type) as type,
   p.modelMag_u, p.modelMag_g, p.modelMag_r, p.
        modelMag_i, p.modelMag_z, p.modelMagErr_u, p.
        modelMagErr_g, p.modelMagErr_r, p.modelMagErr_i,
        p.modelMagerr_z, p.z, p.zErr
FROM #x x, #upload u, SpecPhotoAll p
WHERE u.up_id = x.up_id and x.objID=p.objID
```

[10] Based on this definition, it seems only hits from user program and web browser information are logged.

Table 1: Summary of the 3 SDSS Datasets (approximation only)

Dataset	Num of Tables	Max Table Attributes	Max Table Tuples
CAS	115	approx. 350	approx. 790 million
SkyServer Logs	7	approx. 20	approx. 630 million
Normalized SkyServer Logs	14	approx. 30	approx. 630 million

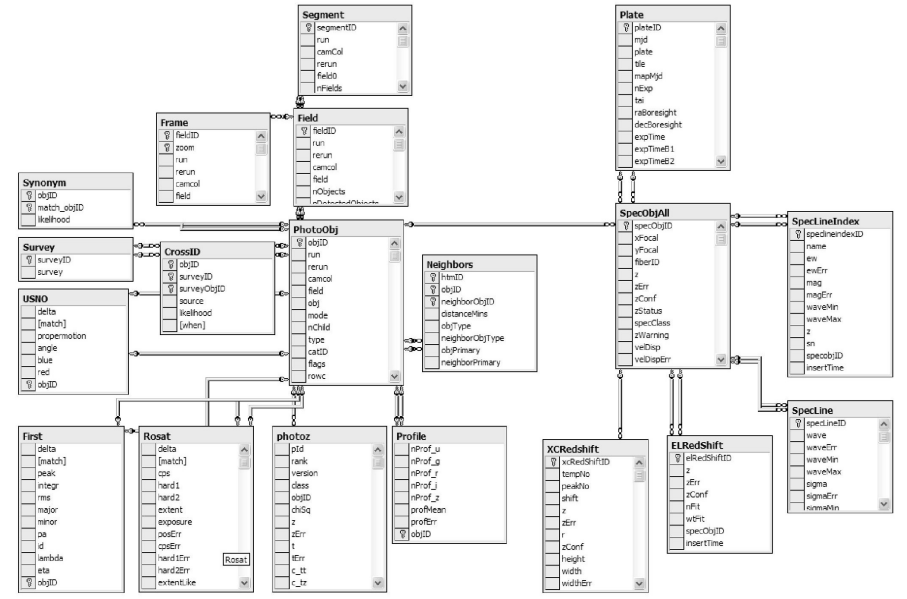

Figure 1: Schema of the Catalog Archive Server (CAS) Database. Image taken from [7]

Table 2: Summary of tracing a session with sessionID 100000 using the normalized Skyserver logs

sessionID	rankInSession	sessionAddressString	theTime	statementID	rows
100000	1	155.198.204.142	2008-04-21,13:04:05.000	48063987	1
100000	2	155.198.204.142	2008-04-21,13:04:06.000	48063987	1
100000	3	155.198.204.142	2008-04-21,13:04:10.000	48063987	1
100000	4	155.198.204.142	2008-04-21,13:04:15.000	48063987	1
100000	5	155.198.204.142	2008-04-21,13:04:15.000	48063987	1
100000	6	155.198.204.142	2008-04-21,13:04:22.000	48063987	1
100000	7	155.198.204.142	2008-04-21,13:04:26.000	48390699	51
100000	8	155.198.204.142	2008-04-21,13:04:34.000	48390699	51
100000	9	155.198.204.142	2008-04-21,13:04:34.000	48063987	1
100000	10	155.198.204.142	2008-04-21,13:04:40.000	48063987	1
100000	11	155.198.204.142	2008-04-21,13:04:58.000	48063987	1

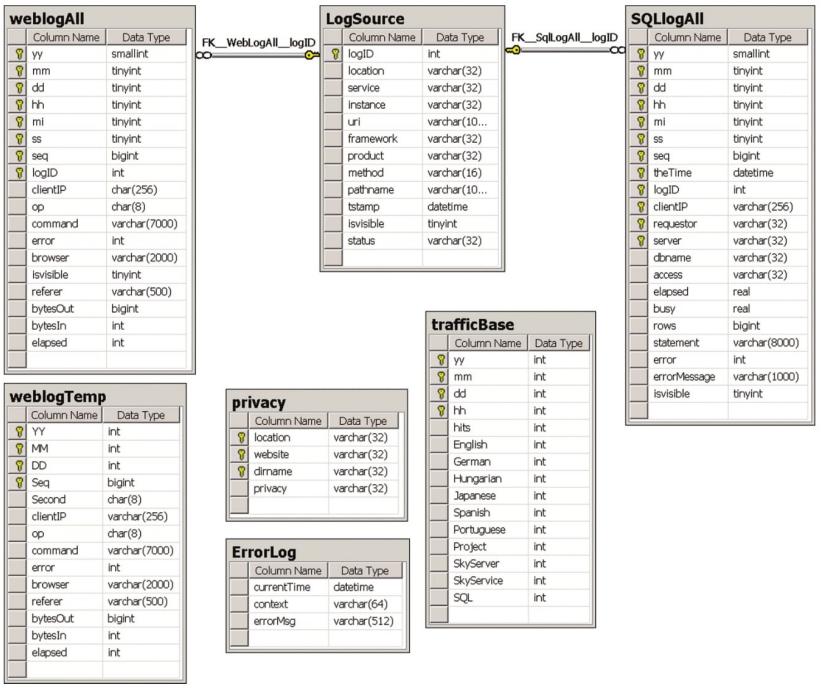

Figure 2: Schema of SkyServer Logs Database. Image taken from [4]

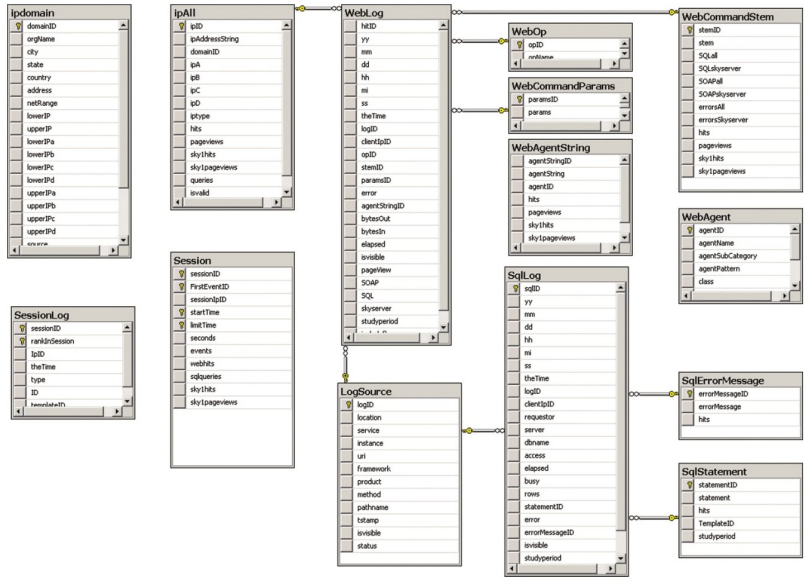

Figure 3: Schema of SkyServer Normalized Logs. Image taken from [4]

```
ORDER BY x.up_id

--statementID 48390699
SELECT u.up_name as name,
   '<a target=INFO href=http://cas.sdss.org/astrodr6/en/
        tools/explore/obj.asp?id=' + cast(x.objId as
        varchar(20)) + '>'+ cast(x.objId as varchar(20))
        + '</a>' as objID, p.ra, p.dec,
   dbo.fPhotoTypeN(p.type) as type,
   p.modelMag_u, p.modelMag_g, p.modelMag_r, p.
        modelMag_i, p.modelMag_z, p.modelMagErr_u, p.
        modelMagErr_g, p.modelMagErr_r, p.modelMagErr_i,
        p.modelMagerr_z
FROM #x x, #upload u, PhotoTag p
WHERE u.up_id = x.up_id and x.objID=p.objID
ORDER BY x.up_id
```

Questions to address :

- How can we group the sql query logs based on the access interface they were submitted through? e.g, CASjobs or SkyServer web-based interface. Do we need to filter the queries to select those that come from the web-based access interface only?
- Can we distinguish between logs based on their agent strings, and should we do this, since, based on the definition of hit, it seems only hits from user programs and web browser information are logged.

How to trace the path that users follow? query the two tables:

- sessionlogs table: combined weblog and Sql log data sorted by client IP address and then time. IDs of logs should be here.
- The sessions table: information about each session, including client IP address, start and end time, and the number of web hits and SQL queries in that session.

2.2 Users vs Sessions

[11] To study SQL usage patterns, we are interested in observing the way human users construct queries, and the way they think and learn. Many large studies have used this heuristic approach and Singh [6] confirmed that 98 percent of all IPs do not send requests after a 30 minutes gap. From the study by Singh [6], there are about 1 million unique IP addresses and 3 million sessions for the Skyserver web interface, and 20,000 unique IP addresses and 100,000 sessions for the SQL interface. The number of unique IP addresses have increased to 3 million 5 years later in the study by Raddick [4]. Raddick observed that while 1 million IP addresses have only one pageview, research-intensive institutions such as UC Berkeley have 10 million pageviews [4]. We are going to use the same heuristic approach to study user SQL usage patterns.

3. RELATED WORK

[12] Many interesting usage characteristics can be drawn from the normalized logs dataset. Singh [6] and Raddick [4] plotted frequency of various items, such as pageviews, against time to observe change of certain patterns over time. In addition, Singh [6] looked at Rank-Size Distribution of SQL query words of the SQL Template provided by SkyServer and observed it corresponds to Zipf's Law, and Raddick observed a faster exponential decrease of frequency as rank decreases. At the same time, using the Rank-Size Distribution of the number of requests per session, Singh observed the Zipfian distribution as well [6]. Hirota [3] performed various statistical tests on the parsed logs such as the K-Means algorithm to partition the query statements in groups and the hierarchical clustering algorithm to cluster the statements. However, no studies observed any pattern of the results returned by submitting the queries.

Since we are interested in obtaining the user-item interaction data of SQL requests, we will be studying the 68 million unique queries in the normalized dataset by Raddick [4]. As outlined in Hirota [3], only queries made through SkyServer, not CASJobs, should be used since all queries in CASJobs are submitted by automated programs; these are unique queries made by human users that are valid and returned tuples. In Hirota's study [2], each item is a token parsed from SQL statements. Tokens are a group of words where each token associates the SQL keywords to table names, column attributes, or other SQL keywords [2]. Examples such as `'select_elliptical'`, `'select_objid'`, `'from_bestobjid'`, `'from_inner'`, `'from_join'`, `'from_photoobj'`, `'where_ra'`, `'where_u'` are some of the tokens parsed from a SQL query statement [3]. Notice the metadata attached to each token, such as attaching 'select' to 'objid' to become `'select_objid'` to indicate that 'objid' is within the 'select' clause. Hirota then converted the parsed queries into feature vectors, where the values in this vector are the frequency of specific tokens [3]. Values of the vectors were transformed using the TF*IDF weighting scheme [3].

4. PROJECT GOALS

For the first part of the project, loading the data into Python class objects, submitting the query to the server as CaJobs, and retrieve the results returned.

Goals in project, for now:

- 21 Sep: Download the data.
- 24 Sep: Re-iterate on this document, address comments, and verify/elaborate ambiguous parts.
- 26 Sep: Trace a single user session and explain what has happened. Explain that user session, use figures if needed.
- 27 Sep: How can we obtain the results for the queries the user submitted?
- 7 Oct: Construct the session-item matrix:
  - Design a scheme for parsing the data, with my help.
  - Write the code, with my help.

5. REFERENCES

[1] Skyserver dr12 sql search. http://cas.sdss.org/dr12/en/tools/search/sql.aspx. Accessed: 2016-09-24.

[2] V. M. Hirota, R. Santos, J. Raddick, and A. Thakar. Mining the sdss skyserver sql queries log. In *SPIE Defense+ Security*, pages 98510S–98510S. International Society for Optics and Photonics, 2016.

[11] I kept this section for now, but it may not be needed

[12] Maybe this section should be called related work.

[3] V. H. Makiyama, M. J. Raddick, and R. D. Santos. Text mining applied to sql queries: A case study for the sdss skyserver. In *2nd Annual International Symposium on Information Management and Big Data*, page 66, 2015.

[4] M. J. Raddick, A. R. Thakar, A. S. Szalay, and R. D. Santos. Ten years of skyserver i: Tracking web and sql e-science usage. *Computing in Science & Engineering*, 16(4):22–31, 2014.

[5] M. J. Raddick, A. R. Thakar, A. S. Szalay, and R. D. Santos. Ten years of skyserver ii: How astronomers and the public have embraced e-science. *Computing in Science & Engineering*, 16(4):32–40, 2014.

[6] V. Singh, J. Gray, A. Thakar, A. S. Szalay, J. Raddick, B. Boroski, S. Lebedeva, and B. Yanny. Skyserver traffic report-the first five years. *arXiv preprint cs/0701173*, 2007.

[7] A. S. Szalay, J. Gray, A. R. Thakar, P. Z. Kunszt, T. Malik, J. Raddick, C. Stoughton, and J. vandenBerg. The sdss skyserver: public access to the sloan digital sky server data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 570–581. ACM, 2002.