Database Practise for Data Science

Sebastian Maneth
University of Edinburgh

October 6, 2015
Agenda

1. Data Conversion
2. SQL Queries
3. Display Results
Relational Databases

Why?

→ fast access to data
→ majority of data stored in relational DBs

Relational DBs

→ very mature pieces of software
  [IBM, Oracle, Microsoft] [PostgreSQL, MySQL, SQLite, etc]

→ SQL standard query language

→ query optimization
1. Data Conversion

→ data typically has to be converted from one format to another before it can be loaded into a Database

→ often you have to write your own code to perform this task
1. Data Conversion

→ data typically has to be converted from one format to another before it can be loaded into a Database

→ often you have to write your own code to perform this task

Example Data

dblp.xml  --  Bibliography data  (computer science)

Size: 1.6GB  (from dblp.uni-trier.de on 01.08.2015)

→ 3.3m entries
→ 1.6m authors
Data Conversion Example

XML → .csv files
eXtensible Markup Language (W3C Standard)
like HTML, but lets you define your own tags
lingua franca for data on the web
uses Unicode

<article mdate="2011-01-11" key="journals/acta/Milner96">
  <author>Robin Milner</author>
  <title>Calculi for Interaction.</title>
  <year>1996</year>
  <pages>707-737</pages>
  <volume>33</volume>
  <journal>Acta Inf.</journal>
  <number>8</number>
  <url>db/journals/acta/acta33.html#Milner96</url>
  <ee>http://dx.doi.org/10.1007/BF03036472</ee>
</article>
Ajeo/dblp-to-csv · GitHub
https://github.com/Ajeo/dblp-to-csv
01.11.2013 - dblp-to-csv - Python parser for dblp xml to csv, with all articles titles and years since 1936.

dblp-to-csv - Python parser for dblp xml to csv, with all ...
java-hackers.com/p/Ajeo/dblp-to-csv/watchers
Python parser for dblp xml to csv, with all articles titles and years since 1936.

xmlutils 1.1 : Python Package Index
https://pypi.python.org/pypi/xmlutils
A set of utilities for processing XML documents and converting to other formats. ... xml files serially, namely converting them to other formats (SQL, CSV, JSON).
import xml.sax
from unidecode import unidecode

def endElement(self, tag):
    if tag == "title":
        self.article['title'] = self.title
    if tag == "year":
        self.article['year'] = self.year
    if len(self.article['title']) > 0 and len(self.article['year']) > 0:
        data = unidecode(self.article['year']+'\n'+self.article['title'])
        self.file.write(data)
        self.article['title'] = ""
        self.article['year'] = ""
    elif self.CurrentData == "dblp":
        self.file.close()
        sys.exit("stop")

def characters(self, content):
    if self.CurrentData == "title":
        self.title = content.strip().rstrip('\n').replace('"','')
    elif self.CurrentData == "year":
        self.year = content.strip().rstrip('\n')

Python code, using SAX-Parser for XML
Issues with this Code:

→ Writes out title/year content whenever another closing tag is encountered

```xml
<article>
<title>Calculi for Interaction.</title>
<year></year>
</article>

<article>
<title></title>
<year>2014</year>
</article>
```
Issues with this Code:

→ converts Unicode characters to ASCII

→ sqlite3 does support Unicode

```
<article>
    <author>Muller</author>
</article>

<article>
    <author>Müller</author>
</article>

→ Muller

→ Müller
Issues with this Code:

→ “characters” is called several times for one text content (by the parser)
1. Data Conversion

LESSON 1

→ always check output manually!
   (before loading it into DB)

→ understand the logic of the converter code!
Example Solution

- open three files: `author.csv`, `paper.csv`, `writtenBy.csv`
- hash author-names by their AID
- if new author encountered, hash it & write it to `author.csv`
- collect authors of one item
- at end of an item, write title/year to `paper.csv`
  write (PID, AID) to `writtenBy.csv` (for all authors of item)
Finally …
2. SQL

→ fire up your favourite Relational Database
→ here: sqlite3

→ we will use only very simple SQL to express interesting queries

Step 1: create tables & indexes

```sql
create table table_name (column_name type [primary key], ... )
```

aka “attribute”

```sql
create index index_name on table_name (column_name, ...);
```
Create Tables & Indexes

$ sqlite3 test.sq3
SQLite version 3.8.11.1 2015-07-29 20:00:57
Enter "help" for usage hints.
sqlite> create table author (aid int primary key,
name text);
sqlite> create table paper (pid int primary key,
title text,
year int);
sqlite> create table writtenBy (pid int,
aid int);
sqlite> create index ai on author (name);
sqlite> create index wi on writtenBy (pid, aid);

sqlite> .separator ";"
sqlite> .import author.csv author
sqlite> .import paper.csv paper
sqlite> .import writtenBy.csv writtenBy
sqlite> .timer on

sqlite3 specific syntax
creates indexes
2. SQL Queries

Aggregates: count, sum, avg, min, max

Conditions: and, or, not, in, <, =, >, like

Combine tables (using set-semantics): union, intersect, except

→ query returns a table

→ where a table is allowed,
  you can place a nested query: (select * from ...)
Text Queries

sqlite> select * 
sqlite> from author 
sqlite> where name="Robin Milner"; 
24;Robin Milner 
Run Time: real 1.174 user 0.672000 sys 0.500000

running time (in seconds)
Text Queries

sqlite> select *
sqlite> from author
sqlite> where name="Robin Milner";
24;Robin Milner
Run Time: real 1.174 user 0.672000 sys 0.500000

with Text Index

sqlite> select *
sqlite> from author
sqlite> where name="Robin Milner";
24;Robin Milner
Run Time: real 0.002 user 0.000000 sys 0.000000

without Text Index
Text Queries

sqlite> select count(*)
sqlite> from paper
sqlite> where title like "%and%"
832133
Run Time: real 6.128 user 4.172000 sys 1.875000
Text Queries

```
sqlite> select count(*)
sqlite> from author a
sqlite> where a.name like "%Milner%";
45
Run Time: real 1.516 user 1.187000 sys 0.250000
```

without Text Index

```
sqlite> select count(*)
sqlite> from author a
sqlite> where a.name like "%Milner%";
45
Run Time: real 1.525 user 1.203000 sys 0.313000
```

with Text Index

 Fallen → use specialized Text Index (e.g. Apache Lucene)
SQL: per author queries

-- title and year of papers by Robin Milner

```sql
select p.title, p.year
from paper p, author a, writtenBy w
where a.name = "Robin Milner"
and a.aid = w.aid and w.pid = p.pid;
```

Calculi for Interaction.;1996
Elements of Interaction – Turing Award Lecture.;1993
An Interview with Robin Milner.;1993
.
.
.
Run Time: real 430.591 user 178.375000 sys 240.688000

without any index
SQL: per author queries

-- title and year of papers by Robin Milner

```sql
select p.title, p.year
from paper p, author a, writtenBy w
where a.name = "Robin Milner"
and a.aid = w.aid and w.pid = p.pid;
```

Calculi for Interaction.;1996
Elements of Interaction – Turing Award Lecture.;1993
An Interview with Robin Milner.;1993

Run Time: real 430.591 user 178.375000 sys 240.688000

without any index

Run Time: real 18.378 user 13.562000 sys 3.485000

with indexes
SQL: per author queries

-- title and year of papers by Robin Milner

```sql
select p.title, p.year
from paper p, author a, writtenBy w
where a.name = "Robin Milner"
and a.aid = w.aid and w.pid = p.pid;
```

Calculi for Interaction.; 1996
Elements of Interaction - Turing Award Lecture.; 1993

Run Time: real 18.378 user 13.562000 sys 3.485000

with indexes

```sql
select p.title, p.year
from paper p, writtenBy w
where w.aid=24 and w.pid = p.pid;
```

Run Time: real 3.579 user 2.484000 sys 1.047000

no join with author
SQL: per author queries

-- title and year of papers by Robin Milner

```sql
select p.title, p.year
from paper p, author a, writtenBy w
where a.name = "Robin Milner"
and a.aid = w.aid and w.pid = p.pid;
```

Calculi for Interaction.;1996
Elements of Interaction – Turing Award Lecture.;1993

.. Run Time: real 18.378 user 13.562000 sys 3.485000

with indexes

```sql
select pid
from writtenBy
where aid=24;
```

Run Time: real 3.659 user 2.484000 sys 1.047000

Oops. – single scan – 
Right index **must** be missing.
create index wi on writtenBy (aid);

select p.title, p.year
from paper p, writtenBy w
where w.aid=24 and w.pid = p.pid;
Calculi for Interaction.;1996
Elements of Interaction - Turing Award Lecture.;1993
..
Run Time: real 0.015 user 0.000000 sys 0.016000

almost 30,000-times faster than without any index
2. SQL Queries

<table>
<thead>
<tr>
<th>LESSON 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ create indexes / primary keys!</td>
</tr>
<tr>
<td>→ avoid joins / use IDs!</td>
</tr>
</tbody>
</table>
2. SQL Queries

How to use “group by”

```
select n, count(n) from T group by n;
```

→ histogram of T

→ without “group by”, semantics of this query is undefined (sqlite3 goes wild, psql reports error)
SQL: per author queries

-- number of papers per year by aid=314 (Jeffrey D. Ullman)

select y.year, y.count
from (select p.year, count(p.year) as count
from (paper p join writtenBy w on (p.pid = w.pid))
where w.aid=314
  group by year
) y;

1966|1
1967|4
1968|8
1969|7
1970|6
1971|4
1972|12

Run Time: real 0.020 user 0.000000 sys 0.031000
SQL: per author queries

-- number of papers per year by aid=314 (Jeffrey D. Ullman)
1966|1
1967|4
1968|8
1969|7
1970|6
1971|4
1972|12
3. Display Results

-- number of papers per year by aid=314 (Jeffrey D. Ullman)
1966|1
1967|4
1968|8
1969|7
1970|6
1971|4
1972|12

→ plot histogram (using R):

.output Ullman.csv
select y.year, y.count ....
.output stdout

$ R
u <- read.csv( 'Ullman.csv', head=FALSE, sep="|" )
barplot( u[,2],u[,1],names.arg=u[,1],col="orange",
main="Jeffrey Ullman's papers per year",las=2 )
3. Display Results

Jeffrey D. Ullman's papers per year

Year: 1966 to 2015
3. Display Results

-- most prolific authors??

```sql
select a.name, count(w.pid) as count
from author a, writtenBy w
Where a.aid = w.aid
group by w.aid
order by count desc limit 40;
```

H. Vincent Poor|1114
Wei Wang|1064
Yan Zhang|999
Wei Liu|981
Wen Gao|926
Philip S. Yu|885
Thomas S. Huang|838
Chin-Chen Chang|795
Lajos Hanzo|790
Elisa Bertino|782
Wei Zhang|779
...
Run Time: real 101.823 user 38.312000 sys 52.125000
Global queries

-- number of papers per year
select y.year, y.count
from (select p.year, count(p.year) as count
      from (paper p join writtenBy w on (p.pid = w.pid))
      group by year) y;
Global queries

-- number of papers per year
select y.year, y.count
from (select p.year, count(p.year) as count
    from (paper p join writtenBy w on (p.pid = w.pid))
group by year) y;
-- number of authors per year
select y.year, y.count
from (select p.year, count(ww.aid) as count
     from (paper p join writtenBy w on (p.pid=w.pid)),writtenBy ww
     where ww.pid=w.pid group by year) y;
SQL: per paper queries

\[
\text{select } \text{avg}(\text{na}) \text{ from (select count(\text{aid}) as \text{na} from writtenBy group by \text{pid})};  \\
2.8366622915623
\]

\[
\text{select max}(\text{na}) \text{ from (select count(\text{aid}) as \text{na} from writtenBy group by \text{pid})};  \\
119
\]

\[
\text{select \text{pid}, count(\text{aid}) as \text{na} from writtenBy group by \text{pid} order by \text{na} \text{ desc limit 10};}  \\
1277503|119  \\
1496761|114  \\
2283215|102  \\
1077916|101  \\
867649|95  \\
2729731|94  \\
587668|86  \\
853657|79  \\
2183911|77  \\
679621|75
\]
SQL: per paper queries

```sql
select avg(na) from
(select count(aid) as na from writtenBy group by pid);
2.8366622915623

select max(na) from
(select count(aid) as na from writtenBy group by pid);
119

select pid,count(aid) as na from writtenBy
group by pid order by na desc limit 10;
1277503|119
1496761|114
2283215|102
1077916|101
867649|95
2729731|94
587668|86
853657|79
2183911|77
679621|75
```

How to get a histogram of this?
SQL: per paper queries

```sql
select pid, count(aid) as na from writtenBy
  group by pid
order by na desc limit 10;
```

<table>
<thead>
<tr>
<th>PID</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1277503</td>
<td>119</td>
</tr>
<tr>
<td>1496761</td>
<td>114</td>
</tr>
<tr>
<td>2283215</td>
<td>102</td>
</tr>
<tr>
<td>1077916</td>
<td>101</td>
</tr>
<tr>
<td>867649</td>
<td>95</td>
</tr>
</tbody>
</table>

---

```sql
select i, 100.0 * count(i) / (select count(*) from paper)
  from CA
  group by i;
```

<table>
<thead>
<tr>
<th>Paper Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18.3%</td>
</tr>
<tr>
<td>2</td>
<td>30.0%</td>
</tr>
<tr>
<td>3</td>
<td>25.3%</td>
</tr>
<tr>
<td>4</td>
<td>14.3%</td>
</tr>
<tr>
<td>5</td>
<td>6.4%</td>
</tr>
<tr>
<td>6</td>
<td>2.7%</td>
</tr>
<tr>
<td>7</td>
<td>1.1%</td>
</tr>
<tr>
<td>8</td>
<td>0.5%</td>
</tr>
<tr>
<td>9</td>
<td>0.3%</td>
</tr>
<tr>
<td>10</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

94.3% of all papers
Who has written the most solo-papers?

-- solo-papers

```
select y.pid from
(select pid, count(aid) as nu from writtenBy group by pid) y
where y.nu=1;
```
Who has written the most solo-papers?

create table SoloPapers (pid int);
create table solo (aid int, pid int);

insert into SoloPapers select y.pid from (select pid,count(aid) as nu from writtenBy group by pid) y where y.nu=1;

insert into solo select w.aid, s.pid from writtenBy w, SoloPapers s where s.pid=w.pid;

select aid,count(pid) as nu from solo group by aid order by nu desc limit 40;
19121;289    552121;166
132395;261    96052;161
192460;258    107288;150
367982;249    487337;148
94760;201     15807;139
584760;186    15644;138
773443;184    ...
Who has written only solo-papers?
- End Lecture 1 -