15-826: Multimedia Databases and Data Mining

Lecture#1: Introduction
Christos Faloutsos
CMU
www.cs.cmu.edu/~christos

Outline

Goal: ‘Find similar / interesting things’
• Intro to DB
• Indexing - similarity search
• Data Mining

Problem

Given a large collection of (multimedia) records, or graphs, find similar/interesting things, ie:
• Allow fast, approximate queries, and
• Find rules/patterns

Q1: Examples, for ‘similar’?
Sample queries

• Similarity search
  – Find pairs of branches with similar sales patterns
  – Find medical cases similar to Smith’s
  – Find pairs of sensor series that move in sync
  – Find shapes like a spark-plug
  – (nn: ‘case based reasoning’)

Problem

Given a large collection of (multimedia) records, or graphs, find similar/interesting things, i.e:

• Allow fast, approximate queries, and
• Find rules/patterns

Q1: Examples, for ‘interesting’?

Sample queries –cont’d

• Rule discovery
  – Clusters (of branches; of sensor data; …)
  – Forecasting (total sales for next year?)
  – Outliers (eg., unexpected part failures; fraud detection)

Example:

Outline

Goal: ‘Find similar / interesting things’

- (crash) intro to DB
- Indexing - similarity search
- Data Mining

Detailed Outline

Intro to DB

- Relational DBMS - what and why?
  - inserting, retrieving and summarizing data
  - views; security/privacy
  - (concurrency control and recovery)

What is the goal of rel. DBMSs
What is the goal of rel. DBMSs

Electronic record-keeping:
Fast and convenient access to information.
Eg.: students, taking classes, obtaining grades;
• find my gpa
• <and other ad-hoc queries>

Main vendors/products

Commercial
• Oracle
• IBM/DB2
• MS SQL-server
• Sybase
• (MS Access, ...

Open source
Postgres (UCB)
mySQL, sqlite,
miniBase (Wisc) (www.sigmod.org)

Detailed Outline

Intro to DB
• Relational DBMS - what and why?
  – inserting, retrieving and summarizing data
  – views; security/privacy
  – (concurrency control and recovery)
How do DBs work?

We use `sqlite3` as an example, from [http://www.sqlite.org](http://www.sqlite.org)

```bash
linux% sqlite3 mydb  # mydb: file
sqlite> create table student ( ssn fixed;
                              name char(20) );
```

```sql
student
<table>
<thead>
<tr>
<th>ssn</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Smith</td>
</tr>
</tbody>
</table>
```

```bash
db> insert into student
    values (123, “Smith”);
```

```sql
takes
<table>
<thead>
<tr>
<th>ssn</th>
<th>c_id</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```bash
db> select * from student;
```

```sql
takes
<table>
<thead>
<tr>
<th>ssn</th>
<th>c_id</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
How do DBs work - cont’d

More than one tables - **joins**

Eg., roster (names only) for 15-826

<table>
<thead>
<tr>
<th>student</th>
<th>takes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn</td>
<td>name</td>
</tr>
<tr>
<td>ssn</td>
<td>c_id</td>
</tr>
</tbody>
</table>

sqlite> select name from student, takes where student.ssn = takes.ssn and takes.c_id = “15826”

---

SQL-DML

General form:

```
select a1, a2, ... an
from r1, r2, ... rm
where P
[order by ...]
[group by ...]
[having ...]
```

---

Aggregation

Find ssn and GPA for each student

<table>
<thead>
<tr>
<th>student</th>
<th>takes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn</td>
<td>name</td>
</tr>
<tr>
<td>ssn</td>
<td>c_id</td>
</tr>
<tr>
<td>123</td>
<td>603</td>
</tr>
<tr>
<td>123</td>
<td>412</td>
</tr>
<tr>
<td>234</td>
<td>603</td>
</tr>
</tbody>
</table>
Aggregation

```
sqlite> select ssn, avg(grade)
    from takes
    group by ssn;
```

<table>
<thead>
<tr>
<th>ssn</th>
<th>c_id</th>
<th>grade</th>
<th>avg(grade)</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>603</td>
<td>4</td>
<td>123</td>
</tr>
<tr>
<td>123</td>
<td>412</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>234</td>
<td>603</td>
<td>3</td>
<td>234</td>
</tr>
</tbody>
</table>

DM!

Detailed Outline

Intro to DB
• Relational DBMS - what and why?
  – inserting, retrieving and summarizing data
  – views; security/privacy
  – (concurrency control and recovery)

Views - what and why?
• suppose you ONLY want to see ssn and GPA (eg., in your data-warehouse)
• suppose secy is only allowed to see GPAs, but not individual grades
• (or, suppose you want to create a short-hand for a query you ask again and again)
• -> VIEWS!

Views

```
sqlite> create view fellowship as (select ssn, avg(grade)
    from takes
    group by ssn);
```
Views

sqlite> create view fellowship as (select ssn, avg(grade) from takes group by ssn);

Views = ‘virtual tables’

sqlite> select * from fellowship;

sqlite> grant select on fellowship to secy;
Detailed Outline

Intro to DB
• Relational DBMS - what and why?
  – inserting, retrieving and summarizing data
  – views; security/privacy
  – (concurrency control and recovery)
• What if slow?
• Conclusions

What if slow?

`sqlite> select * from irs_table where ssn='123';`

Q: What to do, if it takes 2hours?

A: build an index

Q': on what attribute? A: `ssn`

Q'': what syntax? A: `create index`
What if slow - #2?

SQLite> create table friends (p1, p2);

Q: Facebook-style: find the 2-step-away people

sqlite> create table friends (p1, p2);
sqlite> select f1.p1, f2.p2
    from friends f1, friends f2
    where f1.p2 = f2.p1;

Q: too slow – now what?

A: ‘explain’:

DM!

Long answer:

- Check the query optimizer (see, say, Ramakrishnan + Gehrke 3rd edition, chapter15):

Conclusions

• (relational) DBMSs: electronic record keepers
• customize them with `create table` commands
• ask SQL queries to retrieve info

Conclusions cont’d

Data mining practitioner’s guide:
• `create view`, for short-hands / privacy
• `group by` + aggregates
• If a query runs slow:
  – `explain select` – to see what happens
  – `create index` – often speeds up queries

For more info:

• Sqlite3: [www.sqlite.org](http://www.sqlite.org) - @ linux.andrew
• Postgres: also @ linux.andrew
  http://www.postgresql.org/docs/
• Ramakrishnan + Gehrke, 3rd edition
• 15-415 web page, eg,
  – http://www.cs.cmu.edu/~christos/courses/dbms.F15

We assume known:

• B-tree indices
• Hashing
  • www.cs.cmu.edu/~christos/courses/826.S16/FOILS-pdf/030_hashing.pdf
• (also, [Ramakrishnan+Gehrke, ch. 10, ch.11])