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
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')

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)

Outline

Goal: ‘Find similar / interesting things’

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

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

• Flexibility
• data independence (can add new tables; new attributes)
• data sharing/concurrency control
• recovery

Why NOT Databases?
Why NOT Databases?

- Price
- additional expertise (SQL/DBA)
- over-kill for small data sets

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

%sqlite3 mydb  # mydb: file
sql>create table student ( 
  ssn fixed;
  name char(20) );

<table>
<thead>
<tr>
<th>student</th>
<th>ssn</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How do DBs work?

sql>insert into student
values (123, "Smith");
sql>select * from student;

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

How do DBs work?

sql>create table takes
    (ssn fixed,
c_id char(5),
grade fixed));

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

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

sql> 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>name</th>
<th>takes</th>
<th>ssn</th>
<th>c_id</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>123</td>
<td>603</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>123</td>
<td>412</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>234</td>
<td>603</td>
<td>3</td>
</tr>
</tbody>
</table>
Aggregation

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

<table>
<thead>
<tr>
<th>ssn</th>
<th>c_id</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>603</td>
<td>4</td>
</tr>
<tr>
<td>123</td>
<td>412</td>
<td>3</td>
</tr>
<tr>
<td>234</td>
<td>603</td>
<td>3</td>
</tr>
<tr>
<td>ssn</td>
<td>avg(grade)</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>3.5</td>
<td></td>
</tr>
<tr>
<td>234</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

What if slow #2?

```sqlite
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?

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

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

<table>
<thead>
<tr>
<th>takes</th>
<th>c_id</th>
<th>grade</th>
<th>ssn</th>
<th>avg(grade)</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>603</td>
<td>4</td>
<td>123</td>
<td>3.5</td>
</tr>
<tr>
<td>123</td>
<td>412</td>
<td>3</td>
<td>234</td>
<td>3</td>
</tr>
</tbody>
</table>

Views

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

Views = ‘virtual tables’

sql> select * from fellowship;

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

sql> grant select on fellowship to secy;

<table>
<thead>
<tr>
<th>ssn</th>
<th>avg(grade)</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>3.5</td>
</tr>
<tr>
<td>234</td>
<td>3.0</td>
</tr>
</tbody>
</table>
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?
sq1ite> select * from irs_table where
     ssn='123';

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

A: build an index

Q': on what attribute?
Q'': what syntax?
What if slow - #2?

sqlite> create table friends (p1, p2);
Facebook-style: find the 2-step-away people

Q: too slow – now what?

A: `explain`:

sqlite> explain select ....
Long term 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

main advantages over flat files & scripts:
- logical + physical data independence (ie., flexibility of adding new attributes, new tables and indices)
- concurrency control and recovery for free
For more info:

- Sqlite3: www.sqlite.org
- postgres: http://www.postgresql.org/docs/
- Microsoft Access: available on ANDREW clusters (PC)
- Ramakrishna + Gehrke, 3rd edition