SkyServer: An Astronomical Database

Computer Science 105
Boston University
David G. Sullivan, Ph.D.

Sloan Digital Sky Survey (SDSS)

- A database of astronomical data (10s of terabytes)
  - five-year survey mapping 1/4 of the night sky
  - information on 100s of millions of objects
    - stars
    - quasars
    - galaxies
  - a collaboration involving scientists from many countries and institutions
- Part of the International Virtual Observatory
Types of Data in SDSS

- Types of data:
  - images
  - spectrograms: graphs of the amount of light emitted by an objects at different wavelengths
    - can be used to determine:
      - object type
      - temperature
      - chemical composition
      - distance from Earth
  - photometric data: info. measured or computed from images
    - brightness, size, etc.
  - spectrographic data: info. measured or computed from spectrograms
    - redshift, spectral type, etc.

SkyServer

- Provides public access to SDSS data
  [http://skyserver.sdss.org](http://skyserver.sdss.org)

- Includes some good overviews of the relevant astronomy

- It also includes a useful set of tools
    - famous places
    - navigate: pan and zoom through the sky
    - search forms: use menus to specify what you’re looking for
    - SQL search: enter an arbitrary query in SQL
The SDSS Database

- A relational database on Microsoft SQLServer
  - Jim Gray of Microsoft Research worked on the implementation

- The schema was devised in part by:
  - coming up with 20 queries that astronomers might pose
  - refining the schema until these queries could be easily answered
  - example questions:
    - find all objects classified as galaxies in a specified region of the night sky
    - find all asteroids in the database
      - why might this be hard?

SDSS Database Schema

- Three main groups of relations:
  - photo: imaging data
  - spectro: spectroscopic data
  - meta: documentation and metadata (data about the data)

- Can use the schema browser to get info.: http://cas.sdss.org/dr7/en/help/browser/browser.asp
Key Relations in the Photo Group

- **PhotoObjAll**: photometric data for all objects
- **Neighbors**: all pairs of objects that are "near" each other
- **PhotoTag**: contains copies of the most important columns from PhotoObjAll
  - how could using this table speed up queries?
  - why is this table potentially problematic?

Views of a Database

- A database view is like a *virtual* table.
- Rows in a database view may contain:
  - attributes drawn from different tables
  - attributes computed from the attributes stored in the tables
  - only some of the rows in a given table
- Why could views be helpful?
Views Provide Another Layer of Abstraction

views (virtual tables)

<table>
<thead>
<tr>
<th>name</th>
<th>class</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jill Jones</td>
<td>2007</td>
<td>21</td>
</tr>
<tr>
<td>Alan Turing</td>
<td>2010</td>
<td>19</td>
</tr>
</tbody>
</table>

logical representation

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>address</th>
<th>class</th>
<th>dob</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345678</td>
<td>Jill Jones</td>
<td>Warren Towers 100</td>
<td>2007</td>
<td>3/10/85</td>
</tr>
<tr>
<td>25252525</td>
<td>Alan Turing</td>
<td>Student Village A210</td>
<td>2010</td>
<td>2/7/88</td>
</tr>
</tbody>
</table>

physical representation

disks

Views in the Photo Group

- The photo group includes views that make it easier to limit our queries to certain types of objects.
  - stars, galaxies, etc.

- Views enable simpler queries:

  ```sql
  SELECT objID, ...
  FROM Galaxy
  WHERE ...
  ```

  instead of:

  ```sql
  SELECT objID, ...
  FROM PhotoObjAll
  WHERE type = GALAXY AND ...
  ```

- Views don't speed up queries.
  - the SELECT command still accesses the full table(s), and it performs the same operations
Example Query: Find All Galaxies

- Find all objects classified as galaxies within a 1-arcmin radius of the point with coordinates (185, -0.5).

```sql
SELECT G.objID, N.distance
FROM Galaxy AS G,
     dbo.fGetNearbyObjEq(185,-0.5,1) AS N
WHERE G.objID = N.objID
ORDER BY N.distance
```

- Note: `dbo.fGetNearbyObjEq()` is a function that returns a table of all objects within a specified radius of a specified point.
  - if we didn't have this function, what could we use instead?

Seeing Images of Query Results

- To see images of objects obtained, we can use the image-list tool:

- It takes (name, ra, dec) tuples, where name is an identifier of our choosing.

- We can use a query to generate the list of requested images.

- Here's a modified version of our earlier query:

```sql
SELECT G.objID AS name, G.ra, G.dec
FROM Galaxy AS G,
     dbo.fGetNearbyObjEq(185,-0.5,1) AS N
WHERE G.objID = N.objID
ORDER BY N.distance
```

- do not add a semi-colon!
Finding Moving Objects

- For a given portion of the night sky, SDSS takes 5 images, one for each of five color filters.
  - red (r), green (g), ultraviolet (u), infrared (i and z)
  - taken over a 5-minute interval
- Most of the objects in the images (stars, galaxies, etc.) are far enough away that they don't appear to move.
- The movement of closer objects like asteroids is detectable.
  - they can show up as separate "dots" when the three images (r, i, g) used to make the color images are superimposed
- example asteroid:

"Slow-Moving" Objects

- If an object moves slowly enough, the program that processes the raw image data is able to recognize that the different measurements belong to a single object.
  - this includes most asteroids
- In such cases, the movement is recorded in two velocity attributes: rowv and colv
  - total velocity = sqrt(rowv^2 + colv^2)
- To find an object that might be a slow-moving asteroid, check for objects with a total velocity that is high enough.
"Fast-Moving" Objects

- Fast-moving objects appear as streaks:

- These aren't recognized as a single object, so a query to find them needs to examine objects that are close to each other.

Here's a query by Szalay, Gray, et al. for fast-moving asteroids:

```sql
SELECT r.objID AS rID, g.objID AS gID, 
  r.run, r.camcol, 
  r.field as field, g.field as gfield, 
  r.ra AS rRA, r.dec AS rDec, 
  g.ra AS gRA, g.dec AS gDec, 
  "Math (note aas(x) ~ x for x>1)" 
  sqrt(power(r.camcol,2)+power(r.cy-g.cy,2)+power(r.cx-g.cx,2)) * 
  (100*60/ps)) AS distance, 
  dbo.dicsURL(r.objID) AS rURL, 
  dbo.dicsURL(g.objID) AS gURL 
FROM PhotoR r, PhotoG g 
WHERE r.run = g.run AND r.camcol=g.camcol -- same run and camera column 
  and abs(r.field-g.field) =< 1 -- adjacent fields 
  -- the ref selection criteria 
  and (power(r.q,g,2) + power(g.q,g,2)) > 0.111111) -- q/u as ellipticity 
  and r.fibmag < r.fibmag_g 
  and r.fibmag < r.fibmag_r 
  and r.fibmag < r.fibmag_i 
  and r.fibmag < r.fibmag_z 
  and r.parentID = 0
  and r.ishot AS_r > 1.5
  and r.ishot AS_g > 1.5
  -- the green selection criteria 
  and (power(g.q,g,2) + power(g.w,g,2)) > 0.111111 
  and g.fibmag < g.fibmag_r 
  and g.fibmag < g.fibmag_i 
  and g.fibmag < g.fibmag_z 
  and g.parentID = 0
  and g.ishot AS_g > 1.5
  and g.ishot AS_r > 2.0
  -- the match-up of the pair (note aas(x) ~ x for x>1)
  and sqrt(power(r.cx-g.cx,2)+power(r.cy-g.cy,2)+power(r.cz-g.cz,2))((100*60/ps)) < 4.0
  and abs(r.fibmag-r.g.fibmag_g) < 2.0
```
"Fast-Moving" Objects (cont.)

- Despite the complexity of the prior query, it's simpler than previous approaches to finding such asteroids.
  - a colleague of Gray and Szalay wrote a 12-page program that operated on the image data itself
    - it took 3 days to execute
  - it took about a day for G&S to develop the SQL query
    - it executed in about 10 minutes

*Being able to pose questions in a few hours and get answers in a few minutes changes the way one views the data: you can experiment with it almost interactively. When queries take 3 days and hundreds of lines of code, one asks questions cautiously.*

– Gray and Szalay

References

