CS1320
Creating Modern Web and Mobile Applications

Lecture 20:

Database Lab

Objective

- Recall the 6 degrees of Kevin Bacon
- You can do the same thing with CDs
  - Relationships based on multiple artists on one CD
  - Relationships based on multiple artists doing the same song
  - Relationships based on both or on other criteria
- Start with an artist
  - Find all related artists
  - Find all artists related to them, etc.
  - Repeat until nothing changes
  - Output interesting information (your choice):
    - What fraction of all artists are in the set?
    - What is the most prominent artist not in the set?
    - What is the maximum number of links needed?
    - How genre specific are the sets?
    - How many non-singleton sets are there? How many singleton sets?

Helpful Relations

- For MYSQL we have precomputed 2 relations
  - shared_disk(artist1,artist2)
    - Entry if artist1 and artist2 are on the same disk
      - CREATE TABLE shared_disk AS
      - SELECT DISTINCT t1.artistid AS artist1, t2.artistid AS artist2
      - FROM track t1, track t2
      - WHERE t1.diskid = t2.diskid
      - AND t1.artistid != t2.artistid;
  - shared_song(artist1,artist2)
    - Entry if artist1 and artist2 both recorded a song with the same name
      - CREATE TABLE shared_song AS
      - SELECT DISTINCT t1.artistid AS artist1, t2.artistid AS artist2
      - FROM track t1, track t2
      - WHERE t1.name = t2.name
      - AND t1.artistid != t2.artistid;

Helpful Collections

- We created a sharedDisk collection in MongoDB
  - _id : artist name key,
  - value: object with related artist => count

```
let mapshareddisk = function() {
  let tracks = this.tracks;
  for (let i = 0; i < tracks.length; ++i) {
    let t1 = tracks[i];
    if (t1.artist == null) continue;
    let rslt = { };
    let use = false;
    for (let j = 0; j < tracks.length; ++j) {
      let t2 = tracks[ j];
      if (t2.artist == null) continue;
      if (t1.artist != t2.artist) {
        rslt[t2.artist] = 1;
        use = true;
       }
     }
    if (use) emit( t1.artist, rslt )
  }
}


let reducer = function(key,values) {
  let rslt = { };
  for (let v of values) {
    for (k in v) {
      if (rslt[k] == undefined) {
        rslt[k] = v[k];
       }
      else {
        rslt[k] += v[k];
       }
     }
   }


  return rslt;
}


db.cds.mapReduce(mapshareddisk,reducer,{ out: "sharedDisk" });
```

Mechanics

- You should write a node.js program
  - Input (artist name) can be
    - Command line
    - Internal constants (easy to change however) (var INPUT = “nsync”;)
    - REPL (read-eval-print loop)
    - From a web page
  - Access the database as needed
    - Both MongoDB and MySQL databases are available
    - Determine which to use and install appropriate node.js modules
  - Based on what relationship you choose
- Note there are about 1.5M artists total
  - Probably some duplicates (might want to start with multiple)
- Plan your program before implementing it

Database Access

- MongoDB
  - mongodb://bdognom-v2.cs.brown.edu/cdquery
  - User id: cs132, Password: csci1320
  - Collection: cds, sharedDisk
  - npm install mongodb --save
- MySQL
  - mysql://cs132:csci1320@bdognom-v2.cs.brown.edu/cdquery
  - Tables: artist, disk, extended, track, words, shared_disk, shared_song
  - npm install any-db-mysql --save
- There is also a 1% sample database available on both
  - cdquery1
  - Will be faster for use in testing :: **USE THIS FIRST**

Implementation Notes

- Main Routine:
  - Given a set of artists, find all related artists
  - This requires one or more database operations
  - With SQL, might want to create a temporary relation of artists
    - Alternative: very long query
    - Create Table ArtistSet { artistid : char(12) }
    - Insert INTO Table ArtistSet Value ("…")
    - SELECT ? FROM ? WHERE ? AND artistid IN (SELECT * FROM ArtistSet)
- Then apply this routine
  - To initial set
  - To the new entries generated each time

Designers

- Design & implement a web page for this assignment
  - Explain the problem to the user
  - Allow input of an artist
    - Possibly search for artist and select a set of equivalent ones
  - Check for artist validity (provide for this, don’t do it)
  - Provide output page showing results
  - What else might you want
    - Find popular artist not in set?
    - Change relationship criteria
- Can team up with concentrators to produce a full application

Next Time

- Mobile applications