6.5830/6.5831
Introduction to Databases

6.5830 Lecture 1- 9/13/2022
Sam Madden
madden@csail.mit.edu

http://dsg.csail.mit.edu/6.5830
<table>
<thead>
<tr>
<th>Mike Cafarella</th>
<th>Geoffrey Yu</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Mike Cafarella" /></td>
<td><img src="image" alt="Geoffrey Yu" /></td>
</tr>
<tr>
<td>Sam Madden</td>
<td>Anna Zeng</td>
</tr>
<tr>
<td><img src="image" alt="Sam Madden" /></td>
<td><img src="image" alt="Anna Zeng" /></td>
</tr>
</tbody>
</table>

Course Staff
http://dsg.csail.mit.edu/6.5830
Email: 6.5830-staff@mit.edu
Ask questions on Piazza! Use sign up link on website.

Lecturers:
  Sam Madden (madden@csail.mit.edu)
  Michael Cafarella (michjc@csail.mit.edu)

TAs:
  Geoffrey Yu (geoffxy@mit.edu)
  Anna Zeng (annazeng@mit.edu)

Office hours: TBD (weekly)
Textbooks

• Readings in Database Systems
  – http://www.redbook.io

• Rest of readings will be drawn from literature (research papers and web pages)
What is A Database?

• Structured Data Collection
  – Records
  – Relationships

• This class: Database Management Systems (DBMSs)
  – Software systems for storing and querying databases
6.5830 Concepts

• Data modeling / layout

• Declarative querying
  – Query processing
  – Algorithms for accessing and manipulating data

• Consistency / Transactions (“ACID”)

• “Big Data” – scaling to massive volumes, many machines
Interactive Large-Scale Visualization using a GPU Database

Todd Mostak
The Need for Interactive Analytics

• Idea: often need to browse massive data sets

• Browsing is best supported through visualization

⇒ ad-hoc analytics, with millisecond response times
Key insight: GPUs have enough memory that a cluster of them can store substantial amounts of data.

Not an accelerator, but a full blown query processor!

Massive parallelism enables interactive browsing interfaces:
- 4x GPUs can provide > 1 TB/sec of bandwidth
- 12 Tflops compute
- Order of magnitude speedups over CPUs, when data is on GPU

“Shared nothing” arrangement

MapD: GPU Accelerated SQL Database

What is MapD?
- A GPU (Graphics Processing Unit) - accelerated SQL column store database
- Scales to any number of Nvidia GPUs
- A real-time map generator
- Uses GPUs to render point and heatmaps of query results in milliseconds
- A WMS web server
- Can serve out of the box as the backend for a web mapping client, allowing for querying and visualization of billions of features
- Fast and cost-effective
- 4 Nvidia commodity GPUs provide over 12 Teraflops of compute power and nearly 1 TB/sec of memory bandwidth

147,201,658 tweets from Oct 1, 2012 to Nov 6, 2012

Relative intensity of “tornado” on Twitter (with point overlay) from February 29, 2012 to March 1, 2012
Search for “flu” showing outbreak over Southeastern U.S.
Today

• Why database systems?

• User’s perspective:
  – Modeling data
  – Querying data
Zoo Website Features

- Admin interface
  - Edit
  - Add an animal

- Public
  - Pictures & Maps

- Zookeeper
  - Feed times

- 1K animals, 5K URLs, 10 admins, 200 keepers
Animals have names, ages, species
Keepers have names
Cages have cleaning times, buildings
Animals are in 1 cage; cages have multiple animals
Keepers keep multiple cages, cages kept by multiple keepers
Our Zoo

Sam the Salamander
Slimy

Mike the Giraffe
Lanky

Sally the Student
Break

• Questions
  – Are there other ways to represent this zoo data than a collection of tables?
  
  – What are tradeoffs in different representations?
Alternatives to Relations

Hierarchy

cage 1
  mike
    giraffe
    13 yrs
  sally
    student
    1 yr

cage 2
  sam
    salamander
    3 yrs

Graph

animals

mike
  cage1

sally

sam
  cage2
Multiple Tabular Representations Are Possible

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>species</th>
<th>cageno</th>
<th>feedtime</th>
<th>bldg</th>
</tr>
</thead>
<tbody>
<tr>
<td>mike</td>
<td>13</td>
<td>giraffe</td>
<td>1</td>
<td>1:30</td>
<td>1</td>
</tr>
<tr>
<td>sam</td>
<td>3</td>
<td>salamander</td>
<td>2</td>
<td>2:30</td>
<td>2</td>
</tr>
<tr>
<td>sally</td>
<td>1</td>
<td>student</td>
<td>1</td>
<td>1:30</td>
<td>1</td>
</tr>
</tbody>
</table>

Is this a good representation? Why or why not?

Not “Normalized” – repeats data. More in later lectures!
SQL – Structured Query Language

SELECT field1, ..., fieldM
FROM table1, ...
WHERE condition1, ...

INSERT INTO table VALUES (field1, ...)
Names of Giraffes

• Imperative
  
  for each row r in animals
  
  if r.species = ‘giraffe’
  
  output r.name

• Declarative
  
  SELECT r.name FROM animals
  
  WHERE r.species = ‘giraffe’
Cages in Building 32

• Imperative
  
  for each row a in animals
  for each row c in cages
  if a.cageno = c.no and c.bldg = 32
  output a

• Declarative
  
  SELECT a.name FROM animals AS a, cages AS c
  WHERE a.cageno = c.no AND c.bldg = 32
Average Age of Bears

• Declarative

```sql
SELECT AVG(age) FROM animals
WHERE species = 'bear'
```
Complex Queries

Find pairs of animals of the same species and different genders older than 1 year:

```
SELECT a1.name, a2.name
FROM animals as a1, animals as a2
WHERE a1.gender = 'M' and a2.gender = 'F'
AND a1.species = a2.species
AND a1.age > 1 and a2.age > 1
```

“self join”

Find cages with salamanders fed later than the average feedtime of any cage:

```
SELECT cages.cageid
FROM cages, animals
WHERE animals.species = 'salamander'
AND animals.cageid = cages.cageid
AND cages.feedtime >
    (SELECT AVG(feedtime) FROM cages)
```

“nested queries”
Find keepers who keep both students and salamanders:

```
SELECT keeper.name
FROM keeper, cages as c1, cages as c2,
     keeps as k1, keeps as k2, animals as a1, animals as a2
WHERE c1.cageid = k1.cageid AND keeper.keeperid = k1.keeperid
    AND c2.cageid = k2.cageid AND keeper.keeperid = k2.keeperid
    AND a1.species = 'student' AND a2.species = 'salamander'
    AND c1.cageid = a1.cageid AND c2.cageid = a2.cageid
```

Declarative Queries: What not How!

• Many possible procedural plans for a given SQL query

• What else could we do?
  – Sort animals on type
    + good for “bears” query
    - Inserts are slower
  – Store animals table in a hash table or tree (“index”)
SQL → Procedural Plan → Optimized Plan → Compiled Program

Select
Bldg == 2

Join
R1.cageno == R2.cageid

Animals
R1

Cages
R2

Binary search?
SQL → Procedural Plan → Optimized Plan → Compiled Program

Select
Bldg == 2

Join
R1.cageno == R2.cageid

Animals
R1

“Predicate push down”

Cages
R2

Programmer just thinks in terms of table operations, not the order or implementation!
Summary: Database Systems

- Relational Model + Schema Design
- Declarative Queries
- Query Optimization
- Efficient access and updates to data
  - Recoverability
  - Consistency