Research in database systems, theory, and programming languages

~15 students + postdocs
Research Areas

Big data processing in the cloud
• **Theory**: optimal query processing  
  Walter Cai
• **Systems**: Myria, efficient & complex processing at scale, image analytics, DBMS+NN, data summarization
• **Usability**: Cloud SLAs, performance tuning, viz analytics

New Types of DBMSs
• Open World DBMS
• Image & video DBMS
• LightDB: VR/AR/MR DBMS  
  Brandon Haynes

Scientific data management
• Collaborations with scientists & deep involvement with eScience Institute

Databases and programming languages
• DBMS & app co-optimization

Probabilistic Databases  
  Laurel Orr

Causality
Towards Application-Specific Databases
Can we generate customized data stores from application code?
Application Inefficiencies

- Code translated to inefficient queries
- Misplaced computation
- Redundant data loads
- Issuing queries with known results
- Loading unused data
- Missing indexes

78% of fixes took fewer than 5 lines
Max app speedup: 39x

<table>
<thead>
<tr>
<th># stars</th>
<th>Application</th>
<th># issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>22k</td>
<td>Discourse (forum)</td>
<td>85</td>
</tr>
<tr>
<td>1k</td>
<td>Lobster (forum)</td>
<td>45</td>
</tr>
<tr>
<td>49k</td>
<td>Gitlab (collaboration)</td>
<td>23</td>
</tr>
<tr>
<td>13k</td>
<td>Redmine (collaboration)</td>
<td>59</td>
</tr>
<tr>
<td>17k</td>
<td>Spree (E-commerce)</td>
<td>20</td>
</tr>
<tr>
<td>1.7k</td>
<td>ROR Ecommerce</td>
<td>11</td>
</tr>
<tr>
<td>697</td>
<td>Fulcrum (task mgmt)</td>
<td>2</td>
</tr>
<tr>
<td>3.5k</td>
<td>Tracks (task mgmt)</td>
<td>30</td>
</tr>
<tr>
<td>18k</td>
<td>Diaspora (social network)</td>
<td>57</td>
</tr>
<tr>
<td>1.2k</td>
<td>Onebody (social network)</td>
<td>76</td>
</tr>
<tr>
<td>8k</td>
<td>Openstreetmap (map)</td>
<td>4</td>
</tr>
<tr>
<td>1.1k</td>
<td>Fallingfruit (map)</td>
<td>16</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>428</strong></td>
</tr>
</tbody>
</table>
SEARCH

Target code  Proof of translation
SEARCH

PROGRAM SYNTHESIS

Target code  Proof of translation
1. Define semantics of map and reduce

```java
void map(Object key, Point [] value)
{  for(Point p : points)
emit("sumxy", SumXY); }
void reduce(Text key, int [] vs)
{  int SumXY = 0;
  for (Integer val : vs)
    SumXY = SumXY + val;
  emit(key, SumXY); }
```

2. Synthesizer infers spec from source

```java
// sequential implementation
void regress(Point [] points)
{  int SumXY = 0;
  for(Point p : points){
    SumXY += p.x * p.y;
  }
  return SumXY;
}
```

3. Retarget spec to Hadoop

```java
SumXY = reduce(map(points, fn), fr)
fn(x,y) = x * y
fr(v1,v2) = v1 + v2
```

Lifted code can be optimized by Hadoop max 32x speedup
Query Optimizers

Autograders

Application Caches
Full decision procedure exists for conjunctive queries

Simple heuristics can already prove many common cases

Deciding the equality of two arbitrary relational queries is undecidable.

Boris Trakhtenbrot
Constraint Solver
Proof Assistant
Check validity of proofs
Q1 == Q2

Cosette

Proof Assistant
Constraint Solver
Finding counterexamples
Q1 != Q2

Shumo Chu
Daniel Li
Nick Anderson
Workload developer (or the query optimizer) inserts calls to Cuttlefish's API to pick physical operators during execution.

Cuttlefish: A Lightweight Primitive for Online Tuning
Cuttlefish: A Lightweight Primitive for Online Tuning

def loopConvolve(image, filters): ...
def fftConvolve(image, filters): ...
def mmConvolve(image, filters): ...

for image, filters in convolutions:

    start = now()
    result = convolve(image, filters)
    elapsedTime = now() - start

    output result, elapsedTime
def loopConvolve(image, filters): ...
def fftConvolve(image, filters): ...
def mmConvolve(image, filters): ...
tuner = Tuner([loopConvolve, fftConvolve, mmConvolve])

for image, filters in convolutions:
    start = now()
    result = convolve(image, filters)
    elapsedTime = now() - start

    output result, elapsedTime
def loopConvolve(image, filters): ...
def fftConvolve(image, filters): ...
def mmConvolve(image, filters): ...
tuner = Tuner([loopConvolve, fftConvolve, mmConvolve])

for image, filters in convolutions:
    convolve, token = tuner.choose()
    start = now()
    result = convolve(image, filters)
    elapsedTime = now() - start

    output result, elapsedTime
def loopConvolve(image, filters): ...
def fftConvolve(image, filters): ...
def mmConvolve(image, filters): ...
tuner = Tuner([loopConvolve, fftConvolve, mmConvolve])

for image, filters in convolutions:
    convolve, token = tuner.choose()
    start = now()
    result = convolve(image, filters)
    elapsedTime = now() - start
    tuner.observe(token, elapsedTime)
    output result, elapsedTime
Note: Y-axis is Log-scale
Select the id for user "Tom"

Select rows with maximum value for each user.

Calculate running average over id.

```
Select id
From table
Where name = "Tom"
```

```
Select x.id, x.customer, x.total
From PURCHASES x
Join (Select p.customer,
       Max(total)
       From PURCHASES p
       Group By p.customer) y
On y.customer = x.customer
And y.max_total = x.total
```

```
Select a.ord, a.val, Avg(b.val)
From t As a Join t As b
Where b.ord <= a.ord
Group By a.ord, a.val
Order By a.ord
```
Scythe

Chenglong Wang

Input tables

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12/25</td>
</tr>
<tr>
<td>2</td>
<td>11/21</td>
</tr>
<tr>
<td>4</td>
<td>12/24</td>
</tr>
</tbody>
</table>

Output tables

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12/25</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>11/21</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>12/24</td>
<td>20</td>
</tr>
</tbody>
</table>

Stored using specialized data structures

Search for abstract queries

Prune query skeletons

Rank results based on simplicity

Instantiate abstract queries
Scythe

Chenglong Wang

Supported features

• SPJ
• Grouping
• Aggregation
• Subqueries
• Outer join
• Exists
• Union

Benchmark: 193

Scythe: 143

Enum: 92

34x faster on avg.

59% can be answered within 20 seconds
Is there something equivalent to argmax in SQL?

In a more general sense: is there a function that will allow me to find the entire row where a value in Column X is the max value of the column?

If I'm reading your question correctly, the following query should do it (assuming that our column names are a, b, and c and that a is the column that we're maximizing):

```sql
select a, b, c
from table
where a=(select max(a) from table);
```

Of course, if you have more than one row where the column a attains its maximum, then you'll get more than one row back from the query. If you want a unique row back, you can add something like "order by b, c limit 1", or use some other way to rank the rows in which a attains its max.

Stackoverflow dataset
• Posts tagged with #sql, #oracle, #database (430k)
• Posts containing an accepted answer in SQL
• Results: 41k (title, query) pairs

Titles summarize post 80% of the time

Filtered away titles
• My query doesn't work!
• Why is my query slow?
• I hate SQL!
Is there something equivalent to argmax in SQL?

```sql
select a, b, c
from table
where a=(select max(a) from table);
```

<table>
<thead>
<tr>
<th>Model</th>
<th>Naturalness</th>
<th>Informativeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code-NN (Ours)</td>
<td>2.6</td>
<td>1.55</td>
</tr>
<tr>
<td>Nearest neighbor</td>
<td>1.9</td>
<td>1.55</td>
</tr>
<tr>
<td>MOSES</td>
<td>1.76</td>
<td>1.36</td>
</tr>
<tr>
<td>ATTN</td>
<td>2.82</td>
<td>0.93</td>
</tr>
</tbody>
</table>
UWDB Collaborators

UW
• Bill Howe (iSchool)
• Andrew Connolly (Astronomy)
• Aaron Lee (Ophtalmology)
• Ariel Rokem (eScience)
• Emilio Zagheni (Sociology)
• Prog Lang & SW Eng group

Industry
• Adobe
• Huawei
• Intel
• Microsoft
• Teradata