MauveDB: Supporting Model-based User Views in Database Systems

Amol Deshpande               Samuel Madden
• model |ˈmædl|
noun
a simplified description, esp. a mathematical one, of a system or process, to assist in calculations and predictions: a statistical model for predicting the survival rates of endangered species.
Need for models in DBs

• Data that is
  • Imprecise
  • Error prone
  • Incomplete
• Rudimentary database use by Scientists
Example: Wireless Sensor Networks

- Berkeley Motes
- Battery operated
- Radio Transmission of data
- Data Storage/Retrieval from DBMS
- Analysis on Matlab, R, etc.
Model-based Views

User View (uniform at all times)

Model projects from raw readings onto grid

Actual Observations Made at Various Times

<table>
<thead>
<tr>
<th>time</th>
<th>x</th>
<th>y</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>10</td>
<td>19.5</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>20.5</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>10</td>
<td>16.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
<th>x</th>
<th>y</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>20.</td>
</tr>
<tr>
<td>0</td>
<td>15</td>
<td>10</td>
<td>18.</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>8</td>
<td>15.</td>
</tr>
</tbody>
</table>
Advantages

• Allows for transparent changes

• Temporal biases removed

Uniform Grid Based Approximation
MauveDB Supported Models

- Regression and Interpolation
- Dynamic probabilistic and linear dynamical systems
MauveDB System Architecture

User
  sql queries
  query results
  Query Processor
  View Manager
  Storage Manager
  Materialized Views
  Raw Data

Administrator
  model creation/update commands
  Catalog
  View Declarations
  Raw Data Definitions

External data generation tools
Query Processing

- Scan View
- Index View
View Definition
CREATE VIEW
RegView(time[0::1],x[0:9:.1],y[0:9:.1],temp)
AS FIT temp USING time, x, y
BASES 1, x, x^2, y, y^2
FOR EACH time T
TRAINING_DATA SELECT temp, time, x, y
FROM raw-temp-readings
WHERE raw-temp-readings.time = T

Regression based View
CREATE VIEW
IntView(time[0::1],sensorid[::1],temp)
AS INTERPOLATE temp USING time, sensorid
FOR EACH sensorid M
TRAINING_DATA SELECT temp, time, sensorid
FROM raw-temp-readings
WHERE raw-temp-readings.sensorid = M
View Maintenance Strategies

• Materialize the Views
• Always Use Base Data
• Partial Materialization/Caching
• Materialize an Intermediate Representation
Materialize the Views

- Update Views as sensor data comes in.
- Query Execution Latency ↓. Why?
- Traditional query processor used
Always Use Base Data

Partial Materialization/Caching

- Self Explanatory.
Materialize an Intermediate Representation

- Specific to the model used
Regression-based Views

\[ \text{temp}(x, y) = \sum_{i=1}^{k} w_i h_i(x, y) \]

\[ \text{temp}(x, y) = w_1 + w_2 x + w_3 x^2 + w_4 y + w_4 y^2 \]
Regression-based Views

\[ H = \begin{pmatrix} h_1(x_1, y_1) & \cdots & h_k(x_1, y_1) \\ \vdots & \ddots & \vdots \\ h_1(x_m, y_m) & \cdots & h_k(x_m, y_m) \end{pmatrix}, f = \begin{pmatrix} temp_1 \\ \vdots \\ temp_m \end{pmatrix} \] (1)

\[ H^T H \mathbf{w}^* = H^T f \]
Intermediate Representation for Regression Based View

\[ H^T H \mathbf{w}^* = H^T f \]

\[ \langle f \cdot g \rangle = \sum_{i=1}^{m} f(x_i, y_i)g(x_i, y_i) \]

\[
H^T H = \begin{pmatrix}
\langle h_1 \cdot h_1 \rangle & \cdots & \langle h_1 \cdot h_k \rangle \\
\langle h_2 \cdot h_1 \rangle & \cdots & \langle h_2 \cdot h_k \rangle \\
\vdots & \ddots & \vdots \\
\langle h_k \cdot h_1 \rangle & \cdots & \langle h_k \cdot h_k \rangle
\end{pmatrix},
H^T f = \begin{pmatrix}
\langle h_1 \cdot f \rangle \\
\langle h_2 \cdot f \rangle \\
\vdots \\
\langle h_k \cdot f \rangle
\end{pmatrix}
\]
Intermediate Representation for Regression Based View

\[ \text{temp}(x_{m+1}, y_{m+1}) \]

\[ \langle h_1 \bullet h_1 \rangle^{\text{new}} = \langle h_1 \bullet h_1 \rangle^{\text{old}} + h_1(x_{m+1}, y_{m+1})^2 \]

\[ \langle f \bullet g \rangle = \sum_{i=1}^{m} f(x_i, y_i)g(x_i, y_i) \]

Incremental Updates!
Intermediate Representation for Regression Based View

- Properties
  - Smaller in size
  - Incrementally updatable
Interpolation-based view

Query: At what time was the temperature equal to temp'?

No Interpolation

Answer = \{ \}

Linear Interpolation

Answer = \{ T' \}

\[ v_1 = v_0 + (v_3 - v_0) \times \frac{t_3 - t_1}{t_3 - t_0} \]
Intermediate Representation for Interpolation Based View

- Prediction Query
- Search Tree on $t$
- Threshold Query
- Interval Tree on $v$
Contour plot

select * where epoch = 2100
Example Queries

Figure 8: Results of running select avg(temp) group by epoch (i) over the raw data, and (ii) over the interpolation-based view. (iii) shows the percentage of sensors reporting at each epoch.
Example Queries

(i) Computed using raw data

(ii) Computed
Example Queries

(ii) Computed using interpolation-based view
Example Queries

(iii) % of Sensors Reporting

Epoch Number

% of Sensor Reporting

0 20 40 60 80 100

0 500 1000 1500 2000 2500

Non-based view
Complex Query

```sql
SELECT t1.sensorid, t2.sensorid, count(*)
FROM ⟨datatable⟩ t1, ⟨datatable⟩ t2
WHERE abs(t1.temp - t2.temp) < 0.2
    AND t1.epoch = t2.epoch
    AND t1.sensorid < t2.sensorid
GROUP BY t1.sensorid, t2.sensorid
HAVING count(*) > 0.95 * (select count(distinct epoch) from ⟨datatable⟩);
```
## Comparison

<table>
<thead>
<tr>
<th>Materialize (FORCE)</th>
<th>Use Base data (FROMSCRATCH)</th>
<th>Partial Mat (LAZY)</th>
<th>Intermediate repn. (COEFF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recompute t1 &amp; t2 as data arrives</td>
<td>Recompute t1 &amp; t2 as query arrives</td>
<td>Use cache if valid/ else recompute</td>
<td>Recompute t1 &amp; t2 using sufficient stats.</td>
</tr>
</tbody>
</table>
Difference b/w View
Maintenance Strategies

• Performance!
Comparing View Maintenance Strategies

- From Scratch (FROMSCRATCH)
- Using an Intermediate Representation (COEFF)
- Lazy Materialization (LAZY)
- Forced Materialization (FORCE)
Dataset

- Intel Lab Dataset
  - 54-node sensor network
- Raw table - 50k records
- 1k Inserts
- 50 point queries
- 10 average queries
Comparing View Maintenance Strategies

(i) Regression, per Sensor

(ii) Interpolation, per Sensor

(iii) Regression, per Epoch

- FromScratch
- Coeff
- Lazy
- Force

Total Time ($)
Comparing View Maintenance Strategies

(i) Regression, per Sensor

(ii) Interpolation

Total Time (s)

- FromScratch
- Coeff
- Lazy
- Force

Insets | Point Queries | Average Queries

Total Time (s)

Inserts | Point

Tuesday, September 6, 11
Comparing View Maintenance Strategies

![Bar Chart]

- **Total Time (s)**
- **Comparison:**
  - Point Queries
  - Average Queries
  - Insertions
  - Point Queries
  - Average Queries
  - (iii) Regression, per Epoch

- **112.4 s**

*Tuesday, September 6, 11*
Comparing View Maintenance Strategies

Insert performance

[Graph showing total insert time in seconds for different view granularities (10m x 10m, 5m x 5m, 1m x 1m, 0.5m x 0.5m) for 'Coeff' and 'Force' categories. The bars indicate higher performance for smaller granularities.]
Matlab vs MauveDB

![Matlab vs MauveDB](image)

- **Load**
- **Regression**
- **Query**

- **Matlab**
- **MauveDB**

- Relearn time
Conclusion

- Statistical models applied inside the database
- Data independence
- Efficient retrieval of data and good query performance
Future Work

- Support for arbitrary dynamic models
- Continuous Queries
- Active Data Acquisition
Related work

- Predictive Model Markup Language and DB2
- SAS Analytics
- Cougar
- TinyDB