Tuple Bubbles: Learned Tuple Representations for Tunable Approximate Query Processing

Damjan Gjurovski
damjan.gjurovski@cs.rptu.de
RPTU Kaiserslautern-Landau

Sebastian Michel
sebastian.michel@cs.rptu.de
RPTU Kaiserslautern-Landau
Group Tuples into Tuple Bubbles
Create Bubble Summaries

Bayesian Network

Deep Autoregressive Model

(Multidimensional) Histogram
Estimate Query Results from Summaries

SELECT COUNT(*)
FROM N, C, O
WHERE C.BAL > 10
AND O.CLERK = 'C#1'
AND N.N_PK = C.N_PK
AND C.C_PK = O.C_PK
Benefits

- Computation performed over less data
- Faster execution
- Less memory
- Applicable in a distributed environment
Three Core Tasks

1. Organize tuples into Bubbles

2. Create representative but compact bubble summaries

3. Query processing over bubble summaries
Creating Tuple Bubbles

- Organize data from tables into groups:
  - Horizontal partitioning (e.g. primary key)
  - Identify dependencies between tables
  - Similarity-based

- Optimization based on foreign key relationships

- Affects selection of bubbles, estimation accuracy, and execution time

<table>
<thead>
<tr>
<th>o_key</th>
<th>c_key</th>
<th>price</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>22.3</td>
<td>01.03.22</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>75.0</td>
<td>01.03.22</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>444.0</td>
<td>02.03.22</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>399.6</td>
<td>02.03.22</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>400.5</td>
<td>03.03.22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c_key</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>C2</td>
</tr>
<tr>
<td>3</td>
<td>C3</td>
</tr>
<tr>
<td>4</td>
<td>C4</td>
</tr>
</tbody>
</table>
Three Core Tasks

1. Organize tuples into Bubbles

2. Create representative but compact bubble summaries

3. Query processing over bubble summaries
Bubble Summaries

- Create **representative** but **compact** models of the bubbles
- One summary per bubble
- Considered summarization models:
  - Bayesian networks
  - Deep autoregressive models
Summaries as **Bayesian Networks**

- Chow-Liu tree structure learning algorithm
- Probability distribution per node given single parent
- $\text{card}(A_i)^{p+1}$ values per attribute
- Keep $k$ most appearing values and group remaining values into $b$ buckets
- Include primary and foreign keys

<table>
<thead>
<tr>
<th>o_key</th>
<th>c_key</th>
<th>price</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>22.3</td>
<td>01.03.22</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>75.0</td>
<td>01.03.22</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>444.0</td>
<td>02.03.22</td>
</tr>
</tbody>
</table>
Summaries as Deep Autoregressive Models

- Learning over the data
- Estimates density per value conditioned on previous attributes
- Per-attribute lossless compression
- Requires complete join result for training

<table>
<thead>
<tr>
<th>c_key</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>C2</td>
</tr>
<tr>
<td>3</td>
<td>C3</td>
</tr>
<tr>
<td>4</td>
<td>C4</td>
</tr>
</tbody>
</table>

\[
P(c_{\text{key}=2}) \quad \vdots \quad P(name=C2 | c_{\text{key}=2}) \quad \vdots \quad P(name=C3 | c_{\text{key}=2}) \quad \vdots \quad P(name=C4 | c_{\text{key}=2})
\]
Three Core Tasks

1. Organize tuples into Bubbles

2. Create representative but compact bubble summaries

3. **Query processing** over bubble summaries
Query Processing

- Access all bubbles based on query relations
- Combination of bubbles for the query relations
- Estimation over single bubbles:
  - Variable elimination (Bayesian networks)
    \[
    P(p) = \sum_o P(o) \sum_c P(c\mid o) \cdot P(p\mid c) \sum_d P(d\mid p)
    \]
  - Progressive sampling (Bayesian networks and deep autoregressive models)
    \[
    P_o(q) = \prod_{i=1}^4 P(A_i\mid A_{<i})
    \]
Estimating Aggregate Queries

- Variable elimination
  - Final probabilities as $P(A_i|E)$
  - Consider ranges for binned attributes

- Progressive sampling
  - Ordering of attributes affects final probabilities
    - Aggregation attribute can be before query predicates
  - Generate samples for aggregation attribute as query predicates
  - Ranges as samples

```sql
SELECT SUM(c_key) FROM orders WHERE price > 75
```
Join Queries with Bayesian Networks

```
SELECT SUM(price)
FROM customer c, orders o
WHERE c.c_key = o.c_key
AND c.name = 4
AND o.date > '02.03.22'
```

**ORDERS**
- `o_key (o)`
- `c_key (c)`
- `price (p)`
- `date (d)`

**CUSTOMER**
- `c_key (c)`
- `name (n)`
Join Queries with Bayesian Networks

SELECT SUM(price) 
FROM customer c, orders o 
WHERE c.c_key = o.c_key 
AND c.name = 4 
AND o.date > 02.03.22

ORDERS
<table>
<thead>
<tr>
<th>o_key (o)</th>
<th>c_key (c)</th>
<th>price (p)</th>
<th>date (d)</th>
</tr>
</thead>
</table>

CUSTOMER
| c_key (c) | name (n) |
Join Queries with Bayesian Networks

SELECT SUM(price) FROM customer c, orders o WHERE c.c_key = o.c_key AND c.name = 4 AND o.date > 02.03.22

ORDERS
o_key (o)  c_key (c)  price (p)  date (d)

CUSTOMER
  c_key (c)  name (n)
Join Queries with Autoregressive Models

- Identify relevant summaries with respect to the query
- Training performed over the complete join
- Number of models is the number of different joinable bubbles

```
SELECT SUM(price) FROM customer c, orders o WHERE c.c_key = o.c_key AND c.name = 4 AND o.date > 02.03.22
```

TBO_1 and TBC_1

TBO_2 and TBC_1
Experimental Evaluation

- **Setup**
  - NVidia GeForce RTX 2080 Ti GPU ([Autoregressive model](#))
  - Intel Xeon E5-2603 v4 CPU@1.7GHz, 128 GB RAM ([Bayesian network](#); Competitors)

- **Datasets**
  - TPC-H, IMDB, Intel Wireless

- **Competitors**
  - PostgreSQL 14
  - VerdictDB (VDB) *Park et al. SIGMOD 2018*
  - Wander join (WJ) *Li et al. ACM Trans. Database Syst. 2019*
  - KD-Pass *Liang et al. SIGMOD 2021*
  - AQP++ *Peng et al. SIGMOD 2018*
Approach Variants

- Bayesian networks (TB_BN)
  - Horizontal partitioning ($k$) with $i$ bubbles per relation (TB_BN_i)
  - Join on foreign key with one network per join (TB_BN_J)
  - Horizontal partitioning and join on foreign key (TB_BN_J_i)

- Deep autoregressive mode (TB_AR)
  - Horizontal partitioning ($k$) with $i$ bubbles per join (TB_AR_i)
## Execution Time and Memory (TPC-H)

- Considered 150 queries
  - 2-5 joins and 2-5 predicates
- Maximal number of partitions $k$ is 3
- Partitioning threshold $\theta$ is 500,000
- Number of buckets between 60 and 200
- Storing 40 to 100 most common values

<table>
<thead>
<tr>
<th>Approach</th>
<th>Avg. Time (PS/VE)</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostgreSQL</td>
<td>1124.9 ms</td>
<td>1399 MB</td>
</tr>
<tr>
<td>TB_BN</td>
<td>12.01 ms / 58.7 ms</td>
<td>4.8 MB</td>
</tr>
<tr>
<td>TB_BN_1</td>
<td>10.3 ms / 45 ms</td>
<td>4.7 MB</td>
</tr>
<tr>
<td>TB_BN_J</td>
<td>16.4 ms / 160.4 ms</td>
<td>10.8 MB</td>
</tr>
<tr>
<td>TB_BN_J_1</td>
<td>16 ms / 150 ms</td>
<td>17.3 MB</td>
</tr>
<tr>
<td>TB_AR</td>
<td>41.3 ms</td>
<td>36.9 MB</td>
</tr>
<tr>
<td>TB_AR_1</td>
<td>40.1 ms</td>
<td>36.6 MB</td>
</tr>
<tr>
<td>TB_AR_2</td>
<td>40.01 ms</td>
<td>73.2 MB</td>
</tr>
<tr>
<td>TB_AR_3</td>
<td>40.01 ms</td>
<td>109.8 MB</td>
</tr>
<tr>
<td>VDB 10%</td>
<td>96.1 ms</td>
<td>147.5 MB</td>
</tr>
<tr>
<td>VDB 50%</td>
<td>874.1 ms</td>
<td>359 MB</td>
</tr>
<tr>
<td>WJ</td>
<td>143.3 ms</td>
<td>702 MB</td>
</tr>
</tbody>
</table>
Accuracy (TPC-H)

- Considered 150 queries
  - 2-5 joins and 2-5 predicates
- Number of partitions $k$ is 3
- Partitioning threshold $\theta$ is 500 000
- Buckets between 60 and 200
- 40 to 100 most common values
- $q_{error} = \max(\frac{true(q)}{est(q)}, \frac{est(q)}{true(q)})$

<table>
<thead>
<tr>
<th>Approach</th>
<th>Q-error (PS/VE)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median</td>
<td>max</td>
<td>avg</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>TB_BN</td>
<td>3.9 / 3.3</td>
<td>5.3<em>10^4 / 5.3</em>10^4</td>
<td>1064.0 / 1049.4</td>
</tr>
<tr>
<td>TB_BN_1</td>
<td>2.3 / 2.29</td>
<td>5.3<em>10^4 / 3.8</em>10^4</td>
<td>1101.9 / 872.1</td>
</tr>
<tr>
<td>TB_BN_J</td>
<td>1.2 / 1.018</td>
<td>9740.0 / 9740.0</td>
<td>220.1 / 241.5</td>
</tr>
<tr>
<td>TB_BN_J_1</td>
<td>2.02 / 2.006</td>
<td>3.1<em>10^4 / 1.1</em>10^6</td>
<td>861.4 / 1.1*10^4</td>
</tr>
<tr>
<td>TB_AR</td>
<td>1.56</td>
<td>9805</td>
<td>408.29</td>
</tr>
<tr>
<td>TB_AR_2</td>
<td>1.58</td>
<td>4.1*10^5</td>
<td>4013.15</td>
</tr>
<tr>
<td>TB_AR_3</td>
<td>1.36</td>
<td>4.1*10^5</td>
<td>3390.7</td>
</tr>
<tr>
<td>VDB 10%</td>
<td>1.18</td>
<td>1.1*10^10</td>
<td>7.8*10^7</td>
</tr>
<tr>
<td>VDB 50%</td>
<td>1.02</td>
<td>1.1*10^10</td>
<td>7.7*10^7</td>
</tr>
<tr>
<td>WJ</td>
<td>1.1</td>
<td>9.7*10^8</td>
<td>3.2*10^7</td>
</tr>
</tbody>
</table>
Conclusion and Future Work

- Approximate query processing over tuple bubbles
- Group data of relations into bubbles
- Compact but representative summaries
- Estimate results using summaries

Future Work

- Alternatives for bubbles creation
- Combining results from single bubbles
- Other approaches for summarization and combinations of approaches
- Standard operators over tuple bubbles
Thank you for your attention!