Interactive Data Exploration

Extensions

Ali El Adi | Bruno Rubio | Deepak Barua | Hussain Syed

Databases and Information Retrieval Integration Project
Recap – Smart-Drill

- Goal: To discover and summarize interesting group of tuples using smart drill down operator. (keeping in mind the fundamental concepts of drill down).
- Group of tuples can be described as a rule.
  - e.g. (a, b, *, 1000)
## Recap – Smart-Drill

### TABLE I: Initial Summary

<table>
<thead>
<tr>
<th>Store</th>
<th>Product</th>
<th>Region</th>
<th>Count</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>6000</td>
<td>0</td>
</tr>
</tbody>
</table>

### TABLE II: Result After First Smart Drill Down

<table>
<thead>
<tr>
<th>Store</th>
<th>Product</th>
<th>Region</th>
<th>Count</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>6000</td>
<td>0</td>
</tr>
<tr>
<td>Target</td>
<td>bicycles</td>
<td>*</td>
<td>200</td>
<td>2</td>
</tr>
<tr>
<td>*</td>
<td>comforters</td>
<td>MA-3</td>
<td>600</td>
<td>2</td>
</tr>
<tr>
<td>Walmart</td>
<td>*</td>
<td>*</td>
<td>1000</td>
<td>1</td>
</tr>
</tbody>
</table>
Problems we Discussed

- Finding the best marginal rule can be difficult in cases of big data sets.
- So it looks like finding interesting rules is also largely dependent on size of the dataset and in case of very large datasets, sample data is used to get the idea of whole data which may lead to less accuracy than a user desired for initially.
Problems we Discussed

- A system component called sample handler is responsible for creating multiple samples of different parts of the table in memory.
- $\textit{MinSS}$ (minimum Sample Size)
- A higher value of $\textit{minSS}$ increasing accuracy but also increases computation cost.
Extensions

- Sample Handler also decides *Memory Capacity or Budget* we can use for the samples.
- Caching the frequently used samples could be the one big step towards handling this problem.
- Need to add a *third parameter / a check point*
- That can define a threshold at which sample or samples needs to be cached.
Extensions

- For Example → A particular part of the table was accessed more than two times during a particular sampling phase.
- Cache that sample for a specific period of time before dumping it.
- Judging by the user data navigation routine, sample needs to be cached.
- Also the time after which they need to be discarded to make room for new ones.
Extensions

- Image taken from Microsoft best practices cache guidance ➔ Just to give a visual idea. (nothing related to that article)
Extensions

- The biggest advantage → in case of very large tables, the number of hits Sample Handler has to make to the Hard Disk would be largely minimized.
The article already discusses about getting the best possible samples into the memory but caching those frequently grazed parts of the table would be a plus.

Even more interesting part would be if cache threshold can also be user specified 😊.
Extensions

- Next ➔ We have another interesting one. 😊
Scientific Paper

Efficient Cube Construction for Smart City Data
Michael Scriney & Mark Roantree

The polynomial complexity of fully materialized coalesced cubes
What is DWARF

Dwarf is a patented (US Patent 7,133,876) highly compressed structure for computing, storing, and querying Data Cubes. It is a highly compressed structure with reduction reaching 1:1,000,000 depending on the data distribution.

The most important aspect of this patented Dwarf technology is that its data fusion (prefix and suffix redundancy elimination) is discovered and eliminated BEFORE the cube is computed and this explains the dramatic reduction in compute time.

The term “Dwarf” is used in analogy to dwarf stars that have a very large condensed mass but occupy very small space.
Sample Input Data

Legend: ('Country Dimension', 'City Dimension', 'Station Dimension', 'Measure')

(Ireland, 'Dublin', 'Fenian St', '3')
(France, 'Amiens', 'Bd Maignan Larviere', '2')
(Ireland, 'Dublin', 'City Quay', '0')
(France, 'Paris', 'Champs Elysses', '9')
DWARF Cube Data Structure
### DWARF Schema

#### DWARF_Schema

<table>
<thead>
<tr>
<th>id</th>
<th>node_count</th>
<th>cell_count</th>
<th>size_as_mb</th>
<th>entry_node_id</th>
<th>is_cube</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>int</td>
<td>int</td>
<td>int</td>
<td>bool</td>
</tr>
</tbody>
</table>

A: DWARF_Schema column family

#### DWARF_Node

<table>
<thead>
<tr>
<th>id</th>
<th>parentIds</th>
<th>childrenIds</th>
<th>root</th>
<th>schema_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>set&lt; int &gt;</td>
<td>set&lt; int &gt;</td>
<td>boolean</td>
<td>int</td>
</tr>
</tbody>
</table>

B: DWARF_Node schema

#### DWARF_Cell

<table>
<thead>
<tr>
<th>id</th>
<th>key</th>
<th>measure</th>
<th>parentNode</th>
<th>pointerNode</th>
<th>leaf</th>
<th>schema_id</th>
<th>dimension_table_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>text</td>
<td>int</td>
<td>int</td>
<td>int</td>
<td>boolean</td>
<td>int</td>
<td>text</td>
</tr>
</tbody>
</table>

C: DWARF_Cell schema
# Complexity for Data Structure Operations

## Common Data Structure Operations

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Time Complexity</th>
<th>Space Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Worst</td>
</tr>
<tr>
<td></td>
<td>Access</td>
<td>Search</td>
</tr>
<tr>
<td>Array</td>
<td>Θ(1)</td>
<td>Θ(n)</td>
</tr>
<tr>
<td>Stack</td>
<td>Θ(n)</td>
<td>Θ(n)</td>
</tr>
<tr>
<td>Queue</td>
<td>Θ(n)</td>
<td>Θ(n)</td>
</tr>
<tr>
<td>Singly-Linked List</td>
<td>Θ(n)</td>
<td>Θ(n)</td>
</tr>
<tr>
<td>Doubly-Linked List</td>
<td>Θ(n)</td>
<td>Θ(n)</td>
</tr>
<tr>
<td>Skip List</td>
<td>Θ(log(n))</td>
<td>Θ(log(n))</td>
</tr>
<tr>
<td>Hash Table</td>
<td>N/A</td>
<td>Θ(1)</td>
</tr>
<tr>
<td>Binary Search Tree</td>
<td>Θ(log(n))</td>
<td>Θ(log(n))</td>
</tr>
<tr>
<td>Cartesian Tree</td>
<td>N/A</td>
<td>Θ(log(n))</td>
</tr>
<tr>
<td>B-Tree</td>
<td>Θ(log(n))</td>
<td>Θ(log(n))</td>
</tr>
<tr>
<td>Red-Black Tree</td>
<td>Θ(log(n))</td>
<td>Θ(log(n))</td>
</tr>
<tr>
<td>Splay Tree</td>
<td>N/A</td>
<td>Θ(log(n))</td>
</tr>
<tr>
<td>AVL Tree</td>
<td>Θ(log(n))</td>
<td>Θ(log(n))</td>
</tr>
<tr>
<td>KD Tree</td>
<td>Θ(log(n))</td>
<td>Θ(log(n))</td>
</tr>
</tbody>
</table>
Complexity for DWARF Cubic Data structure

\[
\text{Dwarf Computation Time} = O \left( T^{d \log_C T + 1} \frac{1}{(\log_C T)!} \right)
\]

Example

Assume \( c=10\times100 \) \( d=10 \) \( T=100 \)

Dwarf Computation Time = \( O(16666.6666666666666666666667) \)
Most Tree Search Algorithms have $O(\log(n))$ time complexity for search operations.

Taking previous example $n=T=100$

$O(\log (100))=2$
B-Tree

An IDE framework
1. Sample data objects
2. Label sample objects as relevant or irrelevant
3. Incorporate feedback in the user profile (classifier)

Feedback restricted to « interesting » or « not interesting »
Feedback

We may seek more information from the feedback, such as:

• Further explanation on why the sample object is relevant e.g. the decisive **attributes** in some feedback  
  - Help narrow the search.

• **Ratings** of sample objects  
  - Help rank interesting patterns.
An IDE framework
SnapToQuery

Proposes:
- New algorithms to define whether the presented query is intended or not.
- New forms of visual and interactive feedbacks through different devices (e.g. motion control)

Progressive Query Materialization

Emphasis on:
- Checkpoint queries identification
- Reusability of queries

New measures are employed, such as:
- Query Checkpoint for Reuse (QCR)
- Query Result Overlap Ratio (QROR)

- Singh, V., & Jain, S. K. A Progressive Query Materialization for Interactive Data Exploration. *Presented at VLDB workshop on Social Data Analytics and Management (SoDAM)*, Sep 2016.
Closing Comments

IDE family

Smart Drill-Down
Query Steering
AlphaSum
DWARF
AIDE
SnapToQuery
Progressive Query Materialization

... and many more to come
Closing Comments

it’s about the journey
not the destination.