DWH and OLAP

Ali Ugur Guler
Ehsan Ul Haq
Agenda

• What is DWH
  • Why DWH, applications,

• What is OLAP
  • Difference between OLTP and OLAP

• Common Terminologies related to DWH

• How to build DWH and OLAP system
  • ETL, Dimensional Modeling

• Indexing on DWH
  • Why we need indexing
  • Indexing techniques
Motivation for DWH

• Drowning in data but starving for information
• Harnessing the data
• Knowledge Driven Economies
• Knowledge is power but intelligence is absolute power
Why not everyday database systems

• Online Transaction Processing Systems
• Suitable for day to day business operations
• Data is read and most likely to be manipulated in each transaction
• Queries are easy to write and simple
• Examples
  • Point of sale systems
  • Flight reservation systems
  • Learning management systems
But business or stack holders need more insight than just transaction capabilities

• Can OLTP answer such questions?
  • Probably yes, but efficiently?
## OLTP VS OLAP

<table>
<thead>
<tr>
<th></th>
<th>OLTP</th>
<th>OLAP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>users</strong></td>
<td>clerk, IT professional</td>
<td>knowledge worker</td>
</tr>
<tr>
<td><strong>function</strong></td>
<td>day to day operations</td>
<td>decision support</td>
</tr>
<tr>
<td><strong>DB design</strong></td>
<td>application-oriented</td>
<td>subject-oriented</td>
</tr>
<tr>
<td><strong>data</strong></td>
<td>current, up-to-date detailed, flat relational isolated</td>
<td>historical, summarized, multidimensional integrated, consolidated</td>
</tr>
<tr>
<td><strong>usage</strong></td>
<td>repetitive</td>
<td>ad-hoc</td>
</tr>
<tr>
<td><strong>access</strong></td>
<td>read/write</td>
<td>lots of scans</td>
</tr>
<tr>
<td></td>
<td>index/hash on prim. key</td>
<td></td>
</tr>
<tr>
<td><strong>unit of work</strong></td>
<td>short, simple transaction</td>
<td>complex query</td>
</tr>
<tr>
<td><strong># records accessed</strong></td>
<td>tens</td>
<td>millions</td>
</tr>
<tr>
<td><strong>#users</strong></td>
<td>thousands</td>
<td>hundreds</td>
</tr>
<tr>
<td><strong>DB size</strong></td>
<td>100MB-GB</td>
<td>100GB-TB</td>
</tr>
<tr>
<td><strong>metric</strong></td>
<td>transaction throughput</td>
<td>query throughput, response</td>
</tr>
</tbody>
</table>
so a DWH would require

• Execution of simple queries on current and historical data
• Ability to perform what-if analysis
• Visualise the trends in a comprehensible way
DWH

- Collection of diverse data
  - subject oriented
  - aimed at executive, decision maker
  - often a copy of operational data
  - with value-added data (e.g., summaries, history)
  - integrated
  - time-varying
  - non-volatile
DWH

• Collection of tools
  • gathering data
  • cleansing, integrating, ...
  • querying, reporting, analysis
  • data mining
  • monitoring, administering warehouse
DWH: Multitier Architecture

- Metadata
- Monitor & Integrator
- OLAP Server
- Analysis Query Reports Data mining
- Data Sources: Operational DBs
- Extract Transform Load Refresh
- Data Warehouse
- Data Marts
- Tools
How we build a DWH

Multistep process: Extract, Transform, Load (ETL)

• **Data extraction**
  • get data from multiple, heterogeneous, and external sources

• **Data cleaning**
  • detect errors in the data and rectify them when possible

• **Data transformation**
  • convert data from legacy or host format to warehouse format

• **Load**
  • sort, summarize, consolidate, compute views, check integrity, and build indices and partitions

• **Refresh**
  • propagate the updates from the data sources to the warehouse
Building a DWH: Dimensional Modeling

- Cost of products in $ for customers in particular store
Building a DWH: Dimensional Modeling

• A **data warehouse** is based on a multidimensional data model which views data in the form of a data cube

• A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
  
  • **Dimension tables**, such as item (item_name, brand, type), or time (day, week, month, quarter, year)
  
  • **Fact table** contains **measures** (such as dollars_sold) and keys to each of the related dimension tables

• In data warehousing literature, an n-D base cube is called a base cuboid. The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid. The lattice of cuboids forms a data cube.
Building a DWH: Fact Table and dimensions

• Modeling data warehouses: dimensions & measures
  • **Star schema**: A fact table in the middle connected to a set of dimension tables
  • **Snowflake schema**: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
  • **Fact constellations**: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation
Star Schema
Snowflake Schema
Constellation Schema

Customer
- ID
- Name
- Address

SC fact
- C ID
- S ID
- amount

Supplier
- ID
- Name
- Address

SP fact
- P ID
- S ID
- amount

Part
- ID
- Name
- Color
Multidimensional Data: Cube Representations

- Sales volume as a function of product, month, and region

Dimensions: \textit{Product}, \textit{Location}, \textit{Time}
Hierarchical summarization paths

Industry \rightarrow Region \rightarrow Year
Category \rightarrow Country \rightarrow Quarter
Product \rightarrow City \rightarrow Month \rightarrow Week
Office \rightarrow Day

Product \rightarrow Month \rightarrow Region

Dimensions: \textit{Product, Location, Time}
Hierarchical summarization paths

Industry \rightarrow Region \rightarrow Year
Category \rightarrow Country \rightarrow Quarter
Product \rightarrow City \rightarrow Month \rightarrow Week
Office \rightarrow Day

Product \rightarrow Month \rightarrow Region
A Sample Data Cube

Total annual sales of TVs in U.S.A.
Typical Cube Operations

• Roll up (drill-up): summarize data
  • *by climbing up hierarchy or by dimension reduction*

• Drill down (roll down): reverse of roll-up
  • *from higher level summary to lower level summary or detailed data, or introducing new dimensions*

• Slice and dice: *project and select*

• Pivot (rotate):
  • *reorient the cube, visualization, 3D to series of 2D planes*
Roll Up
Slicing
## Pivoting

<table>
<thead>
<tr>
<th>Country</th>
<th>Sales</th>
<th>Date</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1st sem</td>
<td>2nd sem</td>
<td>Total</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ireland</td>
<td>20</td>
<td>23</td>
<td></td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>France</td>
<td>126</td>
<td>138</td>
<td></td>
<td>264</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Germany</td>
<td>56</td>
<td>48</td>
<td></td>
<td>104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>202</td>
<td>209</td>
<td></td>
<td>411</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• OLAP has some major differences with OLTP:
  
  • Infrequent Updates
  
  • Ad-hoc Queries
  
  • Denormalized data
• Is there an indexing technique that would be inefficient for OLTP but very efficient for OLAP?
Bitmap Indexing

• High costs for, inserting, deleting.

• Very efficient for data with low cardinality in read-only mode.

• Very fast Boolean operations

• Null values are retained
• Cardinality, is the measurement of distinct values in a row.

<table>
<thead>
<tr>
<th>Row ID</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rid1</td>
<td>Male</td>
</tr>
<tr>
<td>Rid2</td>
<td>Male</td>
</tr>
<tr>
<td>Rid3</td>
<td>Male</td>
</tr>
<tr>
<td>Rid4</td>
<td>Female</td>
</tr>
<tr>
<td>Rid5</td>
<td>Male</td>
</tr>
</tbody>
</table>

Low Cardinality Example

<table>
<thead>
<tr>
<th>Row ID</th>
<th>Phone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rid1</td>
<td>N1</td>
</tr>
<tr>
<td>Rid2</td>
<td>N2</td>
</tr>
<tr>
<td>Rid3</td>
<td>N3</td>
</tr>
<tr>
<td>Rid4</td>
<td>N4</td>
</tr>
<tr>
<td>Rid5</td>
<td>N5</td>
</tr>
</tbody>
</table>

High Cardinality Example
• Attributes are represented as bit vectors

<table>
<thead>
<tr>
<th>Row ID</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rid1</td>
<td>Male</td>
</tr>
<tr>
<td>Rid2</td>
<td>Male</td>
</tr>
<tr>
<td>Rid3</td>
<td>Male</td>
</tr>
<tr>
<td>Rid4</td>
<td>Female</td>
</tr>
<tr>
<td>Rid5</td>
<td>Male</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rid1</th>
<th>Rid2</th>
<th>Rid3</th>
<th>Rid4</th>
<th>Rid5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Female</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Bitmap representation of a table (Low Cardinality)
• As cardinality goes, columns will increase.

<table>
<thead>
<tr>
<th>Row ID</th>
<th>Phone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rid1</td>
<td>N1</td>
</tr>
<tr>
<td>Rid2</td>
<td>N2</td>
</tr>
<tr>
<td>Rid3</td>
<td>N3</td>
</tr>
<tr>
<td>Rid4</td>
<td>N4</td>
</tr>
<tr>
<td>Rid5</td>
<td>N5</td>
</tr>
</tbody>
</table>

Bitmap representation of a table (High Cardinality)
• Boolean operations are easy

• Result of a Boolean operation with bitmaps are an another bitmap.
"How many of our married customers live in the central or west regions?"

SELECT COUNT(*) FROM customer
WHERE MARITAL_STATUS = 'married' AND REGION IN('central', 'west');

<table>
<thead>
<tr>
<th>Customer No</th>
<th>Marital Status</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Single</td>
<td>East</td>
</tr>
<tr>
<td>102</td>
<td>Married</td>
<td>Central</td>
</tr>
<tr>
<td>103</td>
<td>Married</td>
<td>West</td>
</tr>
<tr>
<td>104</td>
<td>Divorced</td>
<td>West</td>
</tr>
<tr>
<td>105</td>
<td>Single</td>
<td>Central</td>
</tr>
<tr>
<td>106</td>
<td>Married</td>
<td>Central</td>
</tr>
<tr>
<td>Customer No</td>
<td>Marital Status</td>
<td>Region</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>--------</td>
</tr>
<tr>
<td>101</td>
<td>Single</td>
<td>East</td>
</tr>
<tr>
<td>102</td>
<td>Married</td>
<td>Central</td>
</tr>
<tr>
<td>103</td>
<td>Married</td>
<td>West</td>
</tr>
<tr>
<td>104</td>
<td>Divorced</td>
<td>West</td>
</tr>
<tr>
<td>105</td>
<td>Single</td>
<td>Central</td>
</tr>
<tr>
<td>106</td>
<td>Married</td>
<td>Central</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
<th>105</th>
<th>106</th>
</tr>
</thead>
<tbody>
<tr>
<td>West</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Central</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(Central) OR (West)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Married</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Result</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bitmap Compression

• Usually Run Length Encoding is used.

• Just use the location of ones.

  [0000 1001 0001] → [5,8,12]
• Bitmap Compression can be used for Boolean operations.

• [0000 1001 0001] → [5,8,12]
• [0100 0101 0000] → [2,6,8]

  • And Operation
• [0000 0001 0000] → [8]

  • Or operation
• [0100 1101 0001] → [2, 5, 6, 8, 12]
Compression Techniques

- Byte aligned Bitmap Code
- Word Aligned Hybrid
B Tree
• B trees are better when:
  • Cardinality is high,
  • Queries are routine/pre-determined (OLTP)

• Bitmap works better when:
  • Cardinality is low
  • Boolean operations are used frequently
  • Queries are ad-hoc (OLAP)
  • Frequent Updates are not necessary.
Other Indexes used in OLAP

• Join Indexes
  • Store the result of a join operation

• Spatial Index
  • Spatial Indexes are suitable for multidimensional data.
Thank You
References

- Data Warehouse Design Modern Principles and Methodologies, Matteo Golfarelli
- Slides Adapted From:
  - “Data Warehousing and OLAP”, Hector Garcia-Molina, Stanford University
  - “Data Warehousing and OLAP”, Technical University of Munich, Germany
- Bitmap Index vs. B-tree Index: Which and When?, Vivek Sharma, Oracle Articles
- Database Data Warehousing Guide, Oracle Help Center