An Overview of Data Warehousing and OLAP Technology

Presentation by Debojit
Discussion by Ali

Data Warehouse Motivation
- Businesses have a lot of data, operational data and facts.
- Data is usually in different databases and in different physical places.
- Decision makers need to access information (data that has been summarized) virtually on the single site.
- Access needs to be fast regardless of the size of data, and how data's age.

What is decision support
- Decision support systems are a class of computerized information systems that support decision making activities.
- Decision support systems usually require consolidating data from many heterogeneous sources: these might include external sources. -Such as stock market feeds.

What is data warehouse
- Data warehouse is a collection of decision support technologies, aimed at enabling the analysts to make better and faster decisions. It consists of subject-oriented, integrated, time-variant, and non-volatile collection of data.
  - It contains data from different sources.
  - It retains a long history.
  - Changes as new data is added to the repository.

Difference between OLAP and OLTP

<table>
<thead>
<tr>
<th></th>
<th>OLTP</th>
<th>OLAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>Clerk, IT professional</td>
<td>Knowledge worker</td>
</tr>
<tr>
<td>Function</td>
<td>Day to day operations</td>
<td>Decision support</td>
</tr>
<tr>
<td>DB Design</td>
<td>Application-oriented</td>
<td>Subject-oriented</td>
</tr>
<tr>
<td>Data</td>
<td>Current, up-to-date</td>
<td>Historical, summarized, multidimensional, ...</td>
</tr>
<tr>
<td>Usage</td>
<td>Repetitive</td>
<td>Adhoc</td>
</tr>
<tr>
<td>Access</td>
<td>Read/Write</td>
<td>Lots of scans</td>
</tr>
<tr>
<td>Unit of work</td>
<td>Short, simple transaction</td>
<td>Complex query</td>
</tr>
<tr>
<td># rec accessed</td>
<td>Tens</td>
<td>Millions</td>
</tr>
<tr>
<td># users</td>
<td>Thousands</td>
<td>Hundreds</td>
</tr>
<tr>
<td>DB size</td>
<td>100 MB-GB</td>
<td>100 GB-TB</td>
</tr>
<tr>
<td>Metric</td>
<td>Transaction throughput</td>
<td>Query throughput</td>
</tr>
</tbody>
</table>

Why do we separate DW from DB?
- Performance reasons:
  - OLAP requires special data organization that supports multidimensional views.
  - OLAP queries would degrade operational DB.
  - OLAP is read only.
  - No concurrency control and recovery.
- Decision support requires historical data.
- Decision support requires consolidated data.
**Typical OLAP architecture**

- Data sources
- Operational DBs
- Data Warehouse
- OLAP Server
- Metadata
- OLAP & Intelligence
- Extract, Transform, Load
- Data Marts
- Client Tools
- Analysis
- Query Reports
- Data Mining

**Utilities**

- **Data Cleaning**
  - Data Migration: simple transformation rules (replace "gender" with "sex")
  - Data Scrubbing: use domain-specific knowledge (e.g., zip codes) to modify data.
  - Data Auditing: discover rules and relationships (or signal violations thereof).

- **Load**
  - Full load: like one big xact – change from old data to new is atomic.
  - Incremental loading ("refresh") makes sense for big warehouses, but transaction model is more complex.

**Database Design Methodology**

- Most data warehouses use a star schema to represent the multi-dimensional model.
- Each dimension is represented by a dimension-table that describes it.
- A fact-table connects to all dimension-tables with a multiple join. Each tuple in fact-table consists of a pointer to each of the dimension-tables.
- Links between the fact-table in the centre and the dimension-tables form a shape like a star. (Star Schema)

**Star Schema Example**

- Each dimension is represented by one table.

  - Store
  - Customer
  - Unit Sales
  - Dollar sales

**Database Design Methodology (contd.)**

- Each dimension is represented by one table.

  - Un-normalized (introduces redundancy)
  - Ex: (Vancouver, BC, Canada, North America)
    (Victoria, BC, Canada, North America)

  Normalize dimension tables ➔ Snowflake Schema

**Discussion**

- Do you think that star schemas are more useful in data warehouses than in RDBMSs? Why or why not?
## Important considerations for DW servers

- Indexing
- Materialized Views
- Transformation of complex queries
- Parallel processing
- ROLAP/MOLAP servers
- SQL extensions

## Materialized Views

- Materializing summary data can help to accelerate several queries.
- Some of the key challenges:
  - Identify the views to materialize
  - Exploit the materialized views to answer queries
  - Efficiently update the materialized views during load and refresh.

## Metadata requirements

- Administrative metadata
- Source database and their contents
- Back-end and front-end tools
- Definitions of the warehouse schema
- Pre-defined queries and reports
- Data mark locations and contents
- Data refresh and purging policies
- User profiles and user access control policies

## Discussion

- We can use materialized views both in relational databases and in data warehouses. Using materialized views in which one is more crucial? Using materialized views in which one is easier? Why?
Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals

J. Gray, al, Microsoft Research
F. Pellow, al, IBM Research

Outline
- Data analysis
- Visualization and dimension reduction
- The relational representation of N-dimensional data
- What is CUBE
- Summary

Data analysis applications
- Looking for anomalies or unusual patterns.
- Extract statistical information
- Four steps to aggregate data across many dimensions
- Represent the dataset as an N-dimensional space

The sales example

<table>
<thead>
<tr>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Number sold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chevy</td>
<td>1994</td>
<td>Black</td>
<td>50</td>
</tr>
<tr>
<td>Chevy</td>
<td>1994</td>
<td>White</td>
<td>40</td>
</tr>
<tr>
<td>Ford</td>
<td>1994</td>
<td>Black</td>
<td>50</td>
</tr>
<tr>
<td>Ford</td>
<td>1994</td>
<td>White</td>
<td>10</td>
</tr>
<tr>
<td>Chevy</td>
<td>1995</td>
<td>Black</td>
<td>85</td>
</tr>
<tr>
<td>Chevy</td>
<td>1995</td>
<td>White</td>
<td>115</td>
</tr>
<tr>
<td>Ford</td>
<td>1995</td>
<td>Black</td>
<td>85</td>
</tr>
<tr>
<td>Ford</td>
<td>1995</td>
<td>White</td>
<td>75</td>
</tr>
</tbody>
</table>

We have ignored a few columns here such as the date of purchase and the dealer.

“Dimensionality Reduction”
Analyse car sales
- Focus on the role of model, year and color of the cars
- Ignore differences between sales along dimensions of date of sale or car dealership
- As a result, extensive constructs are used, such as cross-tabulation, subtotals, roll-up and drill-down

Problems with SQL
The three most common problems faced by the SQL GROUP BY are:
1. Histograms
2. Roll-up and drill-down
3. Cross Tabulations
Histograms

- Standard SQL does not allow aggregation over computed categories.
- For example, if we had to sort the car sales by type and then perform aggregation functions on it, standard SQL would not support it.

SELECT day, nation, MAX(Temp)
FROM Weather
GROUP BY Day(Time) AS day,
    Nation(Latitude, Longitude) AS nation;

One Dimensional Aggregation

Example: Car sales for year 1994 and 1995 showed in table_1:

<table>
<thead>
<tr>
<th>Model</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chevy</td>
<td>290</td>
</tr>
<tr>
<td>Ford</td>
<td>220</td>
</tr>
</tbody>
</table>

If we need to know the sales for model, we can easily query it by:

SELECT sales
FROM table_1
GROUP BY model

Three Dimensional Aggregation

If we need more dimensional generalization of these operators

Table_2:

<table>
<thead>
<tr>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chevy</td>
<td>1994</td>
<td>black</td>
<td>50</td>
</tr>
<tr>
<td>Chevy</td>
<td>1995</td>
<td>black</td>
<td>85</td>
</tr>
<tr>
<td>Chevy</td>
<td>1994</td>
<td>white</td>
<td>40</td>
</tr>
<tr>
<td>Chevy</td>
<td>1995</td>
<td>white</td>
<td>115</td>
</tr>
</tbody>
</table>

Discussion

- How useful is multi-dimensional aggregation?
- Besides the data warehousing applications mentioned in the paper, can you think of any other application for multi-dimensional aggregation and data cubes?

Roll up/Drill down

If we need to query the sales by model, by year, and by color, then how can we do it?

Typically, we can make a report as showed by Table_3a:

<table>
<thead>
<tr>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Sales by Model</th>
<th>Sales by Year</th>
<th>Sales by Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chevy</td>
<td>1994</td>
<td>black</td>
<td>50</td>
<td>48</td>
<td>92</td>
</tr>
<tr>
<td>Chevy</td>
<td>1995</td>
<td>black</td>
<td>85</td>
<td>83</td>
<td>173</td>
</tr>
<tr>
<td>Chevy</td>
<td>1995</td>
<td>white</td>
<td>115</td>
<td>115</td>
<td>230</td>
</tr>
<tr>
<td>Chevy</td>
<td>1995</td>
<td>white</td>
<td>40</td>
<td>40</td>
<td>200</td>
</tr>
</tbody>
</table>

Roll up/Drill down (contd.)

For Table_3a:

- Concepts: going up the levels is called rolling-up the data.
- Going down is called drilling-down into the data.
- In this table, sales are rolled up by using totals and subtotals.
- Data is aggregated by Model, then by Year, then by Color.
- The report shows data aggregated at three levels, that is, at Model level, Year level, and Color level.
- Data aggregated at each distinct level produces a sub-total.
Problems

What problems with Table_3a approach?

- Table_3a suggests creating 2N aggregation columns for a roll-up of N elements. That is, there are six columns in table_3a.
- Also, the representation of Table_3a is not relational, because the empty cells (presumably NULL values) cannot form a key.

A pivot table in Excel

The approach by using a pivot table in Excel is showed by table_3c:

Table_3c:

<table>
<thead>
<tr>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cherry</td>
<td>1994</td>
<td>Black</td>
<td>30</td>
</tr>
<tr>
<td>Cherry</td>
<td>1994</td>
<td>White</td>
<td>40</td>
</tr>
<tr>
<td>Cherry</td>
<td>1994</td>
<td>ALL</td>
<td>90</td>
</tr>
<tr>
<td>Cherry</td>
<td>1995</td>
<td>Black</td>
<td>83</td>
</tr>
<tr>
<td>Cherry</td>
<td>1995</td>
<td>White</td>
<td>113</td>
</tr>
<tr>
<td>Cherry</td>
<td>1995</td>
<td>ALL</td>
<td>200</td>
</tr>
<tr>
<td>Cherry</td>
<td>ALL</td>
<td>ALL</td>
<td>290</td>
</tr>
</tbody>
</table>

What problems with pivot table approach?

- The pivot operator typically aggregates cells based on values in the cells.
- Pivot creates columns based on subsets of column values—this is a much larger set.
- If one pivots on two columns containing N and M values, the resulting pivot table has N x M values, that's so many columns and such obtuse column names.

ALL value approach

One more approach by adding an ALL value is available:

- Do not extend the result table to have many new columns
- Avoid the exponential growth of columns by overloading column values
- The dummy value "ALL" has been added to fill in the super-aggregation items

ALL value approach (contd.)

For Table_3a:

- This is a 3-dimensional roll-up
- It have three unions
- The fact is that aggregating over N dimensions requires N such UNIONS!

ALL value approach (contd.)

Since table-3a is a relation, it could be built using SQL, like this statement:

```sql
SELECT 'ALL', 'ALL', 'ALL', SUM(Sales) FROM Sales WHERE Model = 'Cherry'
UNION
SELECT Model, 'ALL', 'ALL', SUM(Sales) FROM Sales WHERE Model = 'Cherry'
GROUP BY Model
UNION
SELECT Model, Year, 'ALL', SUM(Sales) FROM Sales WHERE Model = 'Cherry'
GROUP BY Model, Year
UNION
SELECT Model, Year, Color, SUM(Sales) FROM Sales WHERE Model = 'Cherry'
GROUP BY Model, Year, Color;
```

ALL value approach (contd.)

How is ALL value approach?

- Expressing roll-up and cross-tab queries with conventional SQL is daunting! Why?
- A six dimension cross tab requires a 64-way union of 64 different GROUP BY operators to build the underlying representation.
- The resulting representation of aggregation is too complex to analyze for optimization. On most SQL systems this will result in 64 scans of the data, 64 sorts or hashes, and a long wait.
Cross tab

Symmetric aggregation result in a table.

<table>
<thead>
<tr>
<th></th>
<th>1994</th>
<th>1995</th>
<th>Total (ALL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chevy</td>
<td>50</td>
<td>85</td>
<td>135</td>
</tr>
<tr>
<td>White</td>
<td>40</td>
<td>115</td>
<td>155</td>
</tr>
<tr>
<td>Total</td>
<td>90</td>
<td>200</td>
<td>290</td>
</tr>
</tbody>
</table>

Ford sales cross tab

<table>
<thead>
<tr>
<th></th>
<th>1994</th>
<th>1995</th>
<th>Total (ALL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>50</td>
<td>85</td>
<td>135</td>
</tr>
<tr>
<td>White</td>
<td>10</td>
<td>75</td>
<td>85</td>
</tr>
<tr>
<td>Total</td>
<td>60</td>
<td>160</td>
<td>220</td>
</tr>
</tbody>
</table>

The CUBE operator

- N-dimensional generalization of simple aggregate functions
- N-i lower-dimensional aggregates are points, lines, planes, cubes
- Data cube operator builds a table containing all these aggregate values
  - OD data cube: a point.
  - 1D data cube: a line & a point.
  - 2D data cube: a cross tabulation, a plane, two lines, and a point.
  - 3D data cube: a cube with three intersecting 2D cross tabs.

The CUBE operator (contd.)

- For example:
  ```sql
  SELECT Model, Year, Color, SUM(Sales) AS sales
  FROM Sales
  WHERE Model in ['Ford', 'Chevy'] AND Year BETWEEN 1994 AND 1995
  GROUP BY CUBE Model, Year, Color
  ```

  - A relational operator
  - GROUP BY and ROLL UP are degenerate forms of the operator.
  - Aggregates over all <select list> attributes in GROUP BY clause as in standard GROUP BY.
  - It UNIONS in each super-aggregate of global cube—substituting ALL for the aggregation columns.
  - If there are N attributes in the <select list>, there will be 2N - 1 super-aggregate value.
  - The super-aggregates are produced by ROLLUP, like running sum or average.

Summary

- The cube operator generalizes and unifies several common and popular concepts: such as aggregates, group by, histograms, roll-ups and drill-downs and, cross tabs.
- The cube operator is based on a relational representation of aggregate data using the ALL value to denote the set over which each aggregation is computed.
- The data cube is easy to compute for a wide class of functions.
- SQL’s basic set of five aggregate functions needs careful extension to include.

Discussion

- How hard did you find it to understand the CUBE operator?
- As a query writer, would you feel comfortable using it? Or, would you rather use the "solutions" described in the previous slides?

Thanks