Data Warehousing and Decision Support
(mostly using Relational Databases)

CS634
Class 20

Slides based on “Database Management Systems” 3rd ed., Ramakrishnan and Gehrke, Chapter 25

Introduction

- Increasingly, organizations are analyzing current and historical data to identify useful patterns and support business strategies.
- Emphasis is on complex, interactive, exploratory analysis of very large datasets created by integrating data from across all parts of an enterprise.
  - Contrast such Data Warehousing and On-Line Analytic Processing (OLAP) with traditional On-line Transaction Processing (OLTP): mostly long queries, instead of the short update Xacts of OLTP.
  - In past, both were using "structured data" that can be fairly easily loaded into a database.
  - Today, businesses also monitor social media, web clicks, etc., which are not properly structured, hard to put in RDB.

Structured vs. Unstructured Data

- So far, we have been working with structured data
- Structured data:
  - Entities with attributes, each fitting a SQL data type
  - Relationships
  - Each row of data is precious
  - Loads into relational tables, long-term storage
  - Can be huge
- Unstructured data, realm of "big data"
  - Often doesn’t fit into E/R model, too sloppy
  - Each piece of data is not precious – it’s statistical
  - Sometimes just processed and thrown away
  - No permanent specialized repository, maybe saved in files
  - Can be really huge

Data Warehouses using RDB vs. Data Lakes using Hadoop

- Both are ways to hold huge amounts of data
- Data lakes hold "big data", use big data techniques to query and analyze data. Hadoop provides a high-availability scalable distributed systems.
- Big data can be original, uncleaned data, vs. cleaned data for RDB systems.
- A data lake can hold both original and cleaned data. Term "data lake" was invented in 2011, i.e., around same time as release of Hadoop.
- RDB Data warehouse technology ends up with data in a form easily understood by business people.
- Big data is not there yet: usually need “data scientists” to interpret the data, write the queries, or at least new queries.
- Of course this is changing...
- Many big businesses have both a traditional data warehouse and a data lake, load some of same data in both.
  - We may reserve “data warehouse” without adjective to encompass both RDB data warehouses and big-data warehouses that provide user-friendly access methods.

Bigness of Data

Huge data warehouses, all on Teradata systems (hard to find current sizes).

See article

- Biggest DW: Walmart, passed 1TB in 1992, 2.8 PB (petabytes) = 2800 TB in 2008, 30 PB in 2014, 40+ PB in 2017, processing 2.5PB/hour, growing...
- eBay: 9 PB DW in 2013, also has 40 PB of big data, uses Hadoop, etc.
- Apple: multiple PB DW

- Big data:
  - Usually over 50TB, can’t fit on one machine
  - Is judged by "velocity" as well as size
  - Google: processed 24 PB of data per day in 2009, invented Map-Reduce, published 2004

Teradata

- Teradata provides a relational database with ANSI compliant SQL, targeted to data warehouses
- Proprietary, expensive ($millions)
- Uses a shared-nothing architecture on many independent nodes
- Partitioning by rows or (more recently) columns
- Scales up well: add node, add network bandwidth for it
- Now supports Hadoop as well as RDBMS: Teradata Appliance for Hadoop
Three Complementary Trends

- **Data Warehousing**: Consolidate data from many sources in one large repository (relational database or data lake).
  - Loading, periodic synchronization of data.
  - Semantic integration, Data cleaning of data on way in (RDB only so far)
  - Both simple and complex queries and views. (SQL, or programmed)
  - Note: SQL is available on top of big data in most systems
- **OLAP/Multidimensional Analysis**
  - Queries based on spreadsheet-style operations and "multidimensional" view of data. Interactive queries. Look at data from different directions, granularity, etc.
  - Big Data Example: Apache Kylin, originally from eBay, available 2017
- **Data Mining**: Exploratory search for interesting trends and anomalies.
  - Note: BI = Business intelligence, analysis of business information, includes OLAP and data mining

Warehousing Issues

- **Semantic Integration**: When getting data from multiple sources, must eliminate mismatches, e.g., different currencies, schemas.
- **Heterogeneous Sources**: Must access data from a variety of source formats and repositories.
  - Replication capabilities can be exploited here.
- **Load, Refresh, Purge**: Must load data, periodically refresh it, and purge too-old data.
- **Metadata Management**: Must keep track of source (lineage) loading time, and other information for all data in the warehouse.

OLAP: Multidimensional data model

- A way to make complex data understandable by business user, etc.
- Example: sales data
- **Dimensions**: Product, Location, Time
- A **measure** is a numeric value like sales we want to understand in terms of the dimensions
- Example measure: dollar sales value “sales”
- **Example data point** (one row of fact/cube table):
  - \(Sales = 25\) for \(pid=1, timeid=1, locid=1\) is the sum of sales for that day, in that location, for that product
  - \(pid=1\): details in Product table
  - \(locid = 1\): details in Location table
- Note aggregation here: sum of sales is most detailed data (but can have all sales data)

Granularity of Data

- Example of last slide uses time at granularity of days
- Individual transactions (sales at cashier) have been added together to make one row in this table
- Note: “measures” can always be aggregated
- Current hardware can handle more data
- Typical data warehouses hold the original transaction data
- So such a fact table has more columns, for example
  - dateid, timeofday, prodid, storeid, txnid, clerkid, sales, ...
Data Warehouse vs. Data for OLAP

- Current DW fact table is huge, with individual transactions, large number of dimensions
- Can only use a subset of this for OLAP, because of explosion of cells
- Take DW fact table, roll up to days (say), drop less important columns, get much smaller data for OLAP
- Load data into OLAP, another tool.
- Table on pg. 851 is a cube table, not a DW fact table
- Can think of OLAP as a cache of most important aggregates of DW tables

Dimension Hierarchies: OLAP, DW

- For each dimension, the set of values can be organized in a hierarchy:

```
  PRODUCT
    category
    pname

  TIME
    year
    quarter
    month
    date

  LOCATION
    country
    state
    city
```

Schema underlying OLAP, used in RDB DW

```
<table>
<thead>
<tr>
<th>TIME</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>date</td>
</tr>
<tr>
<td>timeid</td>
<td>week</td>
</tr>
<tr>
<td>month</td>
<td>year</td>
</tr>
<tr>
<td>quarter</td>
<td>state</td>
</tr>
<tr>
<td>date</td>
<td>country</td>
</tr>
<tr>
<td>timeid</td>
<td>city</td>
</tr>
<tr>
<td>pid</td>
<td>locid</td>
</tr>
<tr>
<td>timeid</td>
<td>locid</td>
</tr>
<tr>
<td>pid</td>
<td>locid</td>
</tr>
<tr>
<td>timeid</td>
<td>locid</td>
</tr>
<tr>
<td>pid</td>
<td>locid</td>
</tr>
<tr>
<td>timeid</td>
<td>locid</td>
</tr>
</tbody>
</table>
```

- Fact/cube table in BCNF; dimension tables not normalized.
- Dimension tables are small; updates/inserts/deletes are rare. So, anomalies less important than good query performance.
- This kind of schema is very common in DW and OLAP, and is called a star schema; computing the join of all these relations is called a star join.
- Note: in OLAP, this is not what the user sees, it’s hidden underneath
- In DW, this is the basic setup, but usually with more dimensions
- Here only one measure, sales, but can have several

OLAP (and DW) Queries

- Influenced by SQL and by spreadsheets.
- A common operation is to aggregate a measure over one or more dimensions.
  - Find total sales.
  - Find total sales for each city, or for each state.
  - Find top five products ranked by total sales.
- Roll-up: Aggregating at different levels of a dimension hierarchy.
  - E.g., Given total sales by city, we can roll-up to get sales by state.

OLAP Queries: MDX (Multidimensional Expressions)

- Originally a Microsoft SQL Server project, but now supported widely in the OLAP industry: Oracle, SAS, SAP, Teradata on server side, as well as Microsoft. Allows client programs to specify OLAP datasets.
- Example from Wikipedia

```
SELECT
  { [Measures].[Store Sales] } ON COLUMNS,
  { [Date].[2002], [Date].[2003] } ON ROWS
FROM Sales
WHERE ( [Store].[USA].[CA] )
```

- The SELECT clause sets the query axes as the Store Sales member of the Measures dimension, and the 2002 and 2003 members of the Date dimension.
- The WHERE clause indicates that the data source is the Sales cube.
- The WHERE clause defines the “slicer axis” as the California member of the Store dimension.

OLAP Queries

- Drill-down: The inverse of roll-up: go from sum to details that were added up before
  - E.g., Given total sales by state, can drill-down to get total sales by county.
  - Drill down again, see total sales by city
  - E.g., Can also drill-down on different dimension to get total sales by product for each state.
OLAP Queries: cross-tabs

With relational DBs, we are used to tables with column names across the top, rows of data. With OLAP, a spreadsheet-like representation is common, called a cross-tabulation:

- One dimension horizontally
- Another vertically

<table>
<thead>
<tr>
<th></th>
<th>WI</th>
<th>CA</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>63</td>
<td>81</td>
<td>144</td>
</tr>
<tr>
<td>1996</td>
<td>38</td>
<td>107</td>
<td>145</td>
</tr>
<tr>
<td>1997</td>
<td>75</td>
<td>35</td>
<td>110</td>
</tr>
<tr>
<td>Total</td>
<td>176</td>
<td>223</td>
<td>339</td>
</tr>
</tbody>
</table>

OLAP Queries: Pivoting

Example cross-tabulation:

- Pivoting: switching dimensions on axes, or choosing what dimensions to show on axes
- Switching dimensions means pivoting around a point in the upper-left-hand corner
- End up with “1995 1996 1997 Total” across top,
- “WI CA Total” down the side

<table>
<thead>
<tr>
<th></th>
<th>WI</th>
<th>CA</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>63</td>
<td>81</td>
<td>144</td>
</tr>
<tr>
<td>1996</td>
<td>38</td>
<td>107</td>
<td>145</td>
</tr>
<tr>
<td>1997</td>
<td>75</td>
<td>35</td>
<td>110</td>
</tr>
<tr>
<td>Total</td>
<td>176</td>
<td>223</td>
<td>339</td>
</tr>
</tbody>
</table>

SQL Queries for cross-tab entries

The cross-tabulation values can be computed using a collection of SQL queries:

\begin{verbatim}
SELECT SUM(S.sales)
FROM Sales S, Times T, Locations L
WHERE S.timeid = T.timeid
AND S.timeid = L.timeid
GROUP BY T.year, L.state

SELECT SUM(S.sales)
FROM Sales S, Times T
WHERE S.timeid = T.timeid
GROUP BY T.year

SELECT SUM(S.sales)
FROM Sales S, Location L
WHERE S.timeid = L.timeid
GROUP BY L.state
\end{verbatim}

The CUBE Operator

- Generalizing the previous example, if there are k dimensions, we have \(2^k\) possible SQL GROUP BY queries that can be generated through pivoting on a subset of dimensions.

\begin{verbatim}
• Equivalent to rolling up Sales on all eight subsets of the set \{pid, locid, timeid\}; each roll-up corresponds to an SQL query of the form:
  SELECT SUM(S.sales)
  FROM Sales S
  GROUP BY grouping-list
\end{verbatim}

Oracle 10+ supports CUBE queries

\begin{verbatim}
select t.year, s.store_state, sum(dollar_sales)
from salesfact f, times t, store s
where f.time_key = t.time_key and s.store_key = f.store_key
group by cube(t.year, s.store_state);
\end{verbatim}

<table>
<thead>
<tr>
<th>YEAR</th>
<th>STORE_STATE</th>
<th>SUM(DOLLAR_SALES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1994</td>
<td>AZ</td>
<td>17903</td>
</tr>
<tr>
<td>1994</td>
<td>CA</td>
<td>17870</td>
</tr>
<tr>
<td>1994</td>
<td>CA</td>
<td>35684</td>
</tr>
<tr>
<td>1994</td>
<td>CO</td>
<td>787637</td>
</tr>
<tr>
<td>1994</td>
<td>TX</td>
<td>323533</td>
</tr>
</tbody>
</table>

Oracle 11+ supports cross-tabs display

Running on db3 (Oracle version 12):

\begin{verbatim}
SQL> select * from ( 2 select cool, stars from yelp_db.review 3 ) pivot ( 4 count(stars) 5 for stars in ( 2,3,4,5) 6 ) order by cool;
\end{verbatim}

Here is the output:

<table>
<thead>
<tr>
<th>COOL</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>323533</td>
<td>421229</td>
<td>787637</td>
<td>1516269</td>
</tr>
<tr>
<td>1</td>
<td>51358</td>
<td>88168</td>
<td>198705</td>
<td>300811</td>
</tr>
<tr>
<td>2</td>
<td>13812</td>
<td>27798</td>
<td>66019</td>
<td>84758</td>
</tr>
<tr>
<td>3</td>
<td>5116</td>
<td>11690</td>
<td>28468</td>
<td>31867</td>
</tr>
<tr>
<td>4</td>
<td>2455</td>
<td>5979</td>
<td>14690</td>
<td>15452</td>
</tr>
</tbody>
</table>

Oracle 10+ supports CUBE queries

\begin{verbatim}
Oracle 10+ supports CUBE queries
\end{verbatim}

Running on db3 (Oracle version 12):

\begin{verbatim}
SQL> select * from ( 2 select cool, stars from yelp_db.review 3 ) pivot ( 4 count(stars) 5 for stars in ( 2,3,4,5) 6 ) order by cool;
\end{verbatim}

Here is the output:

<table>
<thead>
<tr>
<th>COOL</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>323533</td>
<td>421229</td>
<td>787637</td>
<td>1516269</td>
</tr>
<tr>
<td>1</td>
<td>51358</td>
<td>88168</td>
<td>198705</td>
<td>300811</td>
</tr>
<tr>
<td>2</td>
<td>13812</td>
<td>27798</td>
<td>66019</td>
<td>84758</td>
</tr>
<tr>
<td>3</td>
<td>5116</td>
<td>11690</td>
<td>28468</td>
<td>31867</td>
</tr>
<tr>
<td>4</td>
<td>2455</td>
<td>5979</td>
<td>14690</td>
<td>15452</td>
</tr>
</tbody>
</table>

Oracle 11+ supports cross-tabs display

Running on db3 (Oracle version 12):

\begin{verbatim}
SQL> select * from ( 2 select cool, stars from yelp_db.review 3 ) pivot ( 4 count(stars) 5 for stars in ( 2,3,4,5) 6 ) order by cool;
\end{verbatim}

Here is the output:

<table>
<thead>
<tr>
<th>COOL</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>323533</td>
<td>421229</td>
<td>787637</td>
<td>1516269</td>
</tr>
<tr>
<td>1</td>
<td>51358</td>
<td>88168</td>
<td>198705</td>
<td>300811</td>
</tr>
<tr>
<td>2</td>
<td>13812</td>
<td>27798</td>
<td>66019</td>
<td>84758</td>
</tr>
<tr>
<td>3</td>
<td>5116</td>
<td>11690</td>
<td>28468</td>
<td>31867</td>
</tr>
<tr>
<td>4</td>
<td>2455</td>
<td>5979</td>
<td>14690</td>
<td>15452</td>
</tr>
</tbody>
</table>

Oracle 10+ supports CUBE queries

Running on db3 (Oracle version 12):

\begin{verbatim}
Oracle 10+ supports CUBE queries
\end{verbatim}

Running on db3 (Oracle version 12):

\begin{verbatim}
Oracle 10+ supports CUBE queries
\end{verbatim}

Running on db3 (Oracle version 12):

\begin{verbatim}
Oracle 10+ supports CUBE queries
\end{verbatim}

Running on db3 (Oracle version 12):

\begin{verbatim}
Oracle 10+ supports CUBE queries
\end{verbatim}

Running on db3 (Oracle version 12):

\begin{verbatim}
Oracle 10+ supports CUBE queries
\end{verbatim}

Running on db3 (Oracle version 12):

\begin{verbatim}
Oracle 10+ supports CUBE queries
\end{verbatim}

Running on db3 (Oracle version 12):

\begin{verbatim}
Oracle 10+ supports CUBE queries
\end{verbatim}
DW data → OLAP

- The CUBE query can do the roll-ups on DW data needed for OLAP
- Excel is the champ at OLAP queries
- Look at video
- This video shows pivot tables for a single Excel worksheet
- But Excel can work with database tables: see this longer video
- Pivot tables: drill down, roll up, pivot, …