Miscellaneous Topics, and The Cloud

Topics

- Object Oriented, Object Relational
- Client-server, Parallel, Distributed Systems
- OLAP/Data Warehouses
- Information Retrieval
- Cloud Computing
  - Data centers, Map-reduce, NoSQL Systems
OLAP

- **On-line Analytical Processing**
- **Why?**
  - Exploratory analysis
  - Interactive
  - Different queries than typical SPJ SQL queries
- **Data CUBE**
  - A summary structure used for this purpose
    - E.g. *give me total sales by zipcode; now show me total sales by customer employment category*
  - Much much faster than using SQL queries against the raw data
    - The tables are **huge**

- **Applications:**
  - Sales reporting, Marketing, Forecasting etc etc

---

Data Warehouses

- **A repository of integrated information for querying and analysis purposes**
- **A (usually) stand-alone system that integrates data from everywhere**
  - Read-only, typically not kept up-to-date with the *real* data
  - Geared toward business analytics, data mining etc…
  - HUGE market today

- **Heavily optimized**
  - Specialized query processing and indexing techniques are used
  - High emphasis on pre-computed data structures like summary tables, **data cubes**

- **Analysis cycle:**
  - Extract data from databases with queries, visualize/analyze with desktop tools
  - E.g., **Tableau**
Data Warehouses

Query processing algorithms heavily optimized for these types of schemas

**Many queries of the type:**
Selections on dimension tables (e.g., state = ‘MD’)
Join fact table with dimension tables
Aggregate on a “measure” attribute (e.g., Quantity, TotalPrice)

**For example:**
select c_city, o_year, SUM(quantity) from Fact, Customer, Product
where p_category = ‘Tablet’;

Data Warehouses

Figure 1. Data Warehousing Architecture

Figure 3. A Star Schema.

Figure 4. A Snowflake Schema.
Need Generalized SQL Groupbys

- drill-down and roll-up

<table>
<thead>
<tr>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Sales by Model by Year by Color</th>
<th>Sales by Model by Year</th>
<th>Sales by Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chevy</td>
<td>1994</td>
<td>black</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>white</td>
<td>40</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1995</td>
<td>black</td>
<td>85</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>white</td>
<td>115</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>200</td>
<td></td>
<td>290</td>
</tr>
</tbody>
</table>

Not relational (null values in the keys)

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

Table 4: Sales Summary

<table>
<thead>
<tr>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chevy</td>
<td>1994</td>
<td>black</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>white</td>
<td>40</td>
</tr>
<tr>
<td>Chevy</td>
<td>1994</td>
<td>ALL</td>
<td>90</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>Chevy</td>
<td>1995</td>
<td>ALL</td>
<td>200</td>
</tr>
<tr>
<td>Chevy</td>
<td>ALL</td>
<td>ALL</td>
<td>290</td>
</tr>
</tbody>
</table>

More problems with Groupbys

- roll-up is asymmetric (e.g. does not aggregate by year or by color alone)
- cross-tabulation (spreadsheets)

```
Table 5: Chevy Sales Cross Tab

<table>
<thead>
<tr>
<th>Color</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>40</td>
<td>115</td>
<td>155</td>
</tr>
<tr>
<td>total (ALL)</td>
<td>90</td>
<td>200</td>
<td>290</td>
</tr>
</tbody>
</table>
```

- even if SQL syntax can be devised, a 6D cross-tab requires 64 groupby queries to generate it and 64 scans and sorts of the data

- most of these are not relational expressions but are in many report writers
An Example

<table>
<thead>
<tr>
<th>Make</th>
<th>Year</th>
<th>Color</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chevy</td>
<td>1990</td>
<td>red</td>
<td>5</td>
</tr>
<tr>
<td>Chevy</td>
<td>1990</td>
<td>white</td>
<td>67</td>
</tr>
<tr>
<td>Chevy</td>
<td>1990</td>
<td>blue</td>
<td>62</td>
</tr>
<tr>
<td>Chevy</td>
<td>1991</td>
<td>red</td>
<td>54</td>
</tr>
<tr>
<td>Chevy</td>
<td>1991</td>
<td>white</td>
<td>95</td>
</tr>
<tr>
<td>Chevy</td>
<td>1991</td>
<td>blue</td>
<td>49</td>
</tr>
<tr>
<td>Chevy</td>
<td>1992</td>
<td>red</td>
<td>31</td>
</tr>
<tr>
<td>Chevy</td>
<td>1992</td>
<td>white</td>
<td>54</td>
</tr>
<tr>
<td>Chevy</td>
<td>1992</td>
<td>blue</td>
<td>71</td>
</tr>
<tr>
<td>Ford</td>
<td>1990</td>
<td>red</td>
<td>64</td>
</tr>
<tr>
<td>Ford</td>
<td>1990</td>
<td>white</td>
<td>62</td>
</tr>
<tr>
<td>Ford</td>
<td>1990</td>
<td>blue</td>
<td>63</td>
</tr>
<tr>
<td>Ford</td>
<td>1991</td>
<td>red</td>
<td>52</td>
</tr>
<tr>
<td>Ford</td>
<td>1991</td>
<td>white</td>
<td>9</td>
</tr>
<tr>
<td>Ford</td>
<td>1991</td>
<td>blue</td>
<td>55</td>
</tr>
<tr>
<td>Ford</td>
<td>1992</td>
<td>red</td>
<td>27</td>
</tr>
<tr>
<td>Ford</td>
<td>1992</td>
<td>white</td>
<td>62</td>
</tr>
<tr>
<td>Ford</td>
<td>1992</td>
<td>blue</td>
<td>39</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>ALL</td>
<td>ALL</td>
<td>942</td>
</tr>
<tr>
<td>chevy</td>
<td>ALL</td>
<td>ALL</td>
<td>510</td>
</tr>
<tr>
<td>ford</td>
<td>ALL</td>
<td>ALL</td>
<td>432</td>
</tr>
<tr>
<td>ALL</td>
<td>1990</td>
<td>ALL</td>
<td>343</td>
</tr>
<tr>
<td>ALL</td>
<td>1991</td>
<td>ALL</td>
<td>314</td>
</tr>
<tr>
<td>ALL</td>
<td>1992</td>
<td>ALL</td>
<td>285</td>
</tr>
<tr>
<td>ALL</td>
<td>ALL</td>
<td>red</td>
<td>165</td>
</tr>
<tr>
<td>ALL</td>
<td>ALL</td>
<td>white</td>
<td>273</td>
</tr>
<tr>
<td>ALL</td>
<td>ALL</td>
<td>blue</td>
<td>339</td>
</tr>
<tr>
<td>chevy</td>
<td>1990</td>
<td>ALL</td>
<td>154</td>
</tr>
<tr>
<td>chevy</td>
<td>1991</td>
<td>ALL</td>
<td>199</td>
</tr>
<tr>
<td>chevy</td>
<td>1992</td>
<td>ALL</td>
<td>157</td>
</tr>
<tr>
<td>ford</td>
<td>1990</td>
<td>ALL</td>
<td>169</td>
</tr>
<tr>
<td>ford</td>
<td>1991</td>
<td>ALL</td>
<td>116</td>
</tr>
<tr>
<td>ford</td>
<td>1992</td>
<td>ALL</td>
<td>128</td>
</tr>
<tr>
<td>chevy</td>
<td>ALL</td>
<td>red</td>
<td>91</td>
</tr>
<tr>
<td>chevy</td>
<td>ALL</td>
<td>white</td>
<td>236</td>
</tr>
<tr>
<td>chevy</td>
<td>ALL</td>
<td>blue</td>
<td>183</td>
</tr>
<tr>
<td>ford</td>
<td>ALL</td>
<td>red</td>
<td>144</td>
</tr>
<tr>
<td>ford</td>
<td>ALL</td>
<td>white</td>
<td>133</td>
</tr>
<tr>
<td>ford</td>
<td>ALL</td>
<td>blue</td>
<td>156</td>
</tr>
<tr>
<td>ALL</td>
<td>1990</td>
<td>red</td>
<td>69</td>
</tr>
<tr>
<td>ALL</td>
<td>1990</td>
<td>white</td>
<td>149</td>
</tr>
<tr>
<td>ALL</td>
<td>1990</td>
<td>blue</td>
<td>125</td>
</tr>
<tr>
<td>ALL</td>
<td>1991</td>
<td>red</td>
<td>107</td>
</tr>
<tr>
<td>ALL</td>
<td>1991</td>
<td>white</td>
<td>104</td>
</tr>
<tr>
<td>ALL</td>
<td>1991</td>
<td>blue</td>
<td>104</td>
</tr>
<tr>
<td>ALL</td>
<td>1992</td>
<td>red</td>
<td>59</td>
</tr>
<tr>
<td>ALL</td>
<td>1992</td>
<td>white</td>
<td>116</td>
</tr>
<tr>
<td>ALL</td>
<td>1992</td>
<td>blue</td>
<td>116</td>
</tr>
</tbody>
</table>

CUBE: Aggregate Operator Generalizing Group By
Data Mining

- **Searching for patterns in data**
  - Typically done in data warehouses

- **Association Rules:**
  - When a customer buys X, she also typically buys Y
  - Use ?
    - Move X and Y together in supermarkets
  - A customer buys a lot of shirts
    - Send him a catalogue of shirts
  - Patterns are not always obvious
    - Classic example: It was observed that men tend to buy beer and diapers together (may be an urban legend)

- **Other types of mining**
  - Classification
  - Decision Trees

DataWarehouses

- **Data analytics a major industry right now, and likely to grow in near future**
  - BIG Data !!
  - Extracting (actionable) knowledge from data really critical
    - Especially in real-time

- **Some key technologies:**
  - Parallelism – pretty much required
  - Column-oriented design
    - Lay out the data column-by-column, rather than row-by-row
  - Heavy pre-computation (like Cubes)
  - New types of indexes
    - Focusing on bitmap representations
  - Heavy compression
  - Map-reduce??
Topics

- Object Oriented, Object Relational
- Client-server, Parallel, Distributed Systems
- OLAP/Data Warehouses
- Information Retrieval
- Cloud Computing
  - Data centers, Map-reduce, NoSQL Systems

Information Retrieval

- Relational DB == Structured data

- Information Retrieval == Unstructured data
  - Evolved independently of each other
    - Still very little interaction between the two
  - Goal: Searching within documents
    - Queries are different; typically a list of words, not SQL
  - E.g. Web searching
    - If you just look for documents containing the words, millions of them
      - Mostly useless
  - Ranking:
    - This is the key in IR
    - Many different ways to do it
      - E.g. something that takes into account term frequencies
    - Pagerank (from Google) seems to work best for Web.
**Relevance Ranking Using Terms**

- **TF-IDF** (Term frequency/Inverse Document frequency) ranking:
  - Let \( n(d) = \text{number of terms in the document } d \)
  - \( n(d, t) = \text{number of occurrences of term } t \text{ in the document } d \).
  - Relevance of a document \( d \) to a term \( t \)

\[
TF (d, t) = \log \left( 1 + \frac{n(d, t)}{n(d)} \right)
\]

  - The log factor is to avoid excessive weight to frequent terms

  - Relevance of document to query \( Q \)

\[
r (d, Q) = \sum_{t \in Q} \frac{TF (d, t)}{n(t)}
\]

**PageRank**

- The probability that a random surfer (who follows links randomly) will end up at a particular page

  - **Intuitively**: Higher the probability, the more important the page

- Surfer model:
  - Choose a random page to visit with probability “alpha”
  - If the number of outgoing edges = \( n \), then visit one of those pages with probability \((1 - \text{alpha})/n\)
Cloud Computing: Outline

- Technologies behind cloud computing
  - Data centers
  - Virtualization
  - Programming Framework: Map-reduce
  - Distributed Key-Value Stores
- Some observations about the marketplace

---

Key-Value Stores

- Some Interesting (somewhat old) numbers
  (http://highscalability.com)
  - Twitter: 177M tweets sent on 3/1/2011 (nothing special about the date), 572,000 accounts added on 3/12/2011
  - Dropbox: 1M files saved every 15 mins
  - Stackoverflow: 3M page views a day (Redis for caching)
  - Wordnik: 10 million API Requests a Day on MongoDB and Scala
  - Mollom: Killing Over 373 Million Spams at 100 Requests Per Second (Cassandra)
  - Facebook's New Real-time Messaging System: HBase to Store 135+ Billion Messages a Month
  - Reddit: 270 Million Page Views a Month in May 2010 (Memcache)

- How to support such scale?
  - Databases typically not fast enough
  - Facebook aims for 3-5ms response times
Issues

- Data Consistency, High Availability, and Low Latency hard to guarantee simultaneously
  - Impossible in some cases especially if networks can fail
  - CAP Theorem: Consistency, Availability, Partition tolerance: pick any 2
- Distributed transactions
  - If a transaction spans multiple machines, what to do?
  - Correct solution: Two-phase Commit
    - Multi-round protocol
    - Too high latencies
- Dealing with replication
  - Replication of data is a must
  - How to keep them updated?
    - Eager vs lazy replication
    - Significant impact on consistency and availability
- Many systems in this space sacrifice consistency

Systems

- Numerous systems designed in last 10 years that look very similar
  - Differences often subtle, and if not hard to pin down, hard to understand
  - Often the differences are about the implementations
- Often called key-value stores
  - The main provided functionality is that of a hashtable
- Some earlier solutions
  - Still very popular
    - Memcached + MySQL + Sharding
      - Sharding == partitioning
      - Store data in MySQL -- use Memcached to cache the data
    - Memcached not really a database, just a cache
    - All kinds of consistency issues
    - But... very very fast
**Systems**

- **MySQL + Memcached: End of an era? (High Scalability Blog)**
  - “If you look at the early days of this blog, when web scalability was still in its heady bloom of youth, many of the articles had to do with leveraging MySQL and memcached. Exciting times. Shard MySQL to handle high write loads, cache objects in memcached to handle high read loads, and then write a lot of glue code to make it all work together. That was state of the art, that was how it was done. The architecture of many major sites still follow this pattern today, largely because with enough elbow grease, it works.”

- Digg moved to Cassandra in 2009; LinkedIn to Voldemort

- Twitter moved to Cassandra recently
  - “.. the rate of growth is accelerating.. a system in place based on shared mysql + memcache .. quickly becoming prohibitively costly (in terms of manpower) to operate.

**Systems**

- **Tokyo, Redis**
  - Very efficient key value stores

- **BigTable (Google), HBase (Apache open source), Cassandra (original Facebook, open sourced), Voldemort (originally LinkedIn)**...
  - At least in original iterations, focused on performance
  - Cassandra later developed more sophisticated {em tunable} consistency (maybe others too)

- **PNUTS (Yahoo!)**
  - Focus on geographically distributed stuff
    - Easier to deal with some issues if we assume everything is a single data center
    - Support tunable consistency for reads: read-any, read-latest etc..
  - Form of master-slave replication
  - No real support for multi-record transactions
Systems

- **Megastore (Google)**
  - Built on top of BigTable -- powers Google App Engine
  - Full ACID using Paxos, replication, two-phase commit
  - Supports notion of “entity groups”
    - e.g., all emails of a user is a single entity group
    - Transactions that span a single entity group are generally fine
    - Transactions that span multiple entity groups would use two-phase commit -- not preferred

- **MongoDB**
  - Perhaps the poster child of key-value NoSQL stores
  - Very scalable
    - Document-oriented storage with JSON-style documents
    - JSON becoming more popular than XML as the interchange format
  - Very loose consistency guarantees

In Summary…

- **Three key pieces of cloud computing**
  - Data centers
    - Increasingly growing in numbers
    - Many challenges in building them, maintaining them etc..
  - Virtualization
  - Programming frameworks
    - Simplest (to explain): just use the virtual machines directly
      - But much harder to manage
    - Using Hadoop or HBase (as appropriate) simplifies the programming quite a bit
      - But Hadoop is open source, and managing hadoop installations not much easier

- **Still many technical challenges to be solved**