Data Warehousing and OLAP

Hector Garcia-Molina
Stanford University

Warehousing

- Growing industry: $8 billion in 1998
- Range from desktop to huge:
  - Walmart: 900-CPU, 2,700 disk, 23TB Teradata system
- Lots of buzzwords, hype
  - slice & dice, rollup, MOLAP, pivot, ...
What is a Warehouse?

- 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
What is a Warehouse?

- Collection of tools
  - gathering data
  - cleansing, integrating, ...
  - querying, reporting, analysis
  - data mining
  - monitoring, administering warehouse

Warehouse Architecture
Why a Warehouse?

- Two Approaches:
  - Query-Driven (Lazy)
  - Warehouse (Eager)

Query-Driven Approach
Advantages of Warehousing

- High query performance
- Queries not visible outside warehouse
- Local processing at sources unaffected
- Can operate when sources unavailable
- Can query data not stored in a DBMS
- Extra information at warehouse
  - Modify, summarize (store aggregates)
  - Add historical information

Advantages of Query-Driven

- No need to copy data
  - less storage
  - no need to purchase data
- More up-to-date data
- Query needs can be unknown
- Only query interface needed at sources
- May be less draining on sources
OLTP vs. OLAP

- **OLTP**: On Line Transaction Processing
  - Describes processing at operational sites
- **OLAP**: On Line Analytical Processing
  - Describes processing at warehouse

### OLTP
- Mostly updates
- Many small transactions
- Mb-Tb of data
- Raw data
- Clerical users
- Up-to-date data
- Consistency, recoverability critical

### OLAP
- Mostly reads
- Queries long, complex
- Gb-Tb of data
- Summarized, consolidated data
- Decision-makers, analysts as users
Data Warehouse

Query / Reporting
What happened?

OLAP
How and/or why did it happen?

Predictive Analytics
What's going to happen?
Business Intelligence (BI)
Data Marts

- Smaller warehouses
- Spans part of organization
  - e.g., marketing (customers, products, sales)
- Do not require enterprise-wide consensus
  - but long term integration problems?

Warehouse Models & Operators

- Data Models
  - relations
  - stars & snowflakes
  - cubes
- Operators
  - slice & dice
  - roll-up, drill down
  - pivoting
  - other
Star Schema
Terms

- Fact table
- Dimension tables
- Measures

Dimension Hierarchies

- store
  - city
  - region

- snowflake schema
- constellations
### Cube

#### Fact table view:

<table>
<thead>
<tr>
<th>sale</th>
<th>prodId</th>
<th>storeId</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>c1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>c1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c3</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>c2</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

#### Multi-dimensional cube:

```
<table>
<thead>
<tr>
<th>c1</th>
<th>c2</th>
<th>c3</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>12</td>
<td>50</td>
</tr>
<tr>
<td>p2</td>
<td>11</td>
<td>8</td>
</tr>
</tbody>
</table>
```
dimensions = 2

### 3-D Cube

#### Fact table view:

<table>
<thead>
<tr>
<th>sale</th>
<th>prodId</th>
<th>storeId</th>
<th>date</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>c1</td>
<td>1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>c1</td>
<td>1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c3</td>
<td>1</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>c2</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c1</td>
<td>2</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

#### Multi-dimensional cube:

```
<table>
<thead>
<tr>
<th>c1</th>
<th>c2</th>
<th>c3</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>44</td>
<td>4</td>
</tr>
<tr>
<td>p2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
dimensions = 3
ROLAP vs. MOLAP

- ROLAP: Relational On-Line Analytical Processing
- MOLAP: Multi-Dimensional On-Line Analytical Processing

Aggregates

- Add up amounts for day 1
- In SQL: `SELECT sum(amt) FROM SALE WHERE date = 1`

```
sale  prodlid storeld date  amt
p1    c1     1    12
p2    c1     1    11
p1    c3     1    50
p2    c2     1    8
p1    c1     2    44
p1    c2     2    4
```

81
### Aggregates

- Add up amounts by day
- In SQL:  
  ```sql
  SELECT date, sum(amt) FROM SALE
  GROUP BY date
  ```

<table>
<thead>
<tr>
<th>sale</th>
<th>prold</th>
<th>storeId</th>
<th>date</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>c1</td>
<td>1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>c1</td>
<td>1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c3</td>
<td>1</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>c2</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c1</td>
<td>2</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ans</th>
<th>date</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>81</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>48</td>
</tr>
</tbody>
</table>

### Another Example

- Add up amounts by day, product
- In SQL:  
  ```sql
  SELECT date, sum(amt) FROM SALE
  GROUP BY date, prodId
  ```

<table>
<thead>
<tr>
<th>sale</th>
<th>prold</th>
<th>date</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>c1</td>
<td>1</td>
<td>62</td>
</tr>
<tr>
<td>p2</td>
<td>c1</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>p1</td>
<td>c2</td>
<td>2</td>
<td>48</td>
</tr>
</tbody>
</table>

Directions:  
- Rollup: 
  ```sql
  SELECT prold, date, sum(amt) FROM SALE
  GROUP BY prold, date
  ```

- Drill-down: 
  ```sql
  SELECT * FROM SALE
  ```
Aggregates

- Operators: sum, count, max, min, median, ave
- “Having” clause
- Using dimension hierarchy
  - average by region (within store)
  - maximum by month (within date)

Cube Aggregation

Example: computing sums
Cube Operators

Extended Cube
Aggregation Using Hierarchies

(customer c1 in Region A; customers c2, c3 in Region B)

Pivoting

Fact table view:

<table>
<thead>
<tr>
<th>sale</th>
<th>prodId</th>
<th>storeId</th>
<th>date</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>c1</td>
<td>1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c3</td>
<td>1</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>c2</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c1</td>
<td>2</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Multi-dimensional cube:
Implementing a Warehouse

- *Monitoring*: Sending data from sources
- *Integrating*: Loading, cleansing,...
- *Processing*: Query processing, indexing, ...
- *Managing*: Metadata, Design, ...

--

Monitoring

- Source Types: relational, flat file, IMS, VSAM, IDMS, WWW, news-wire, ...
- Incremental vs. Refresh

<table>
<thead>
<tr>
<th>customer</th>
<th>id</th>
<th>name</th>
<th>address</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>53</td>
<td>joe</td>
<td>10 main</td>
<td>sfo</td>
</tr>
<tr>
<td></td>
<td>81</td>
<td>fred</td>
<td>12 main</td>
<td>sfo</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>sally</td>
<td>80 willow</td>
<td>la</td>
</tr>
</tbody>
</table>
Monitoring Techniques

- Periodic snapshots
- Database triggers
- Log shipping
- Data shipping (replication service)
- Transaction shipping
- Polling (queries to source)
- Screen scraping
- Application level monitoring

Advantages & Disadvantages!!

Monitoring Issues

- Frequency
  - periodic: daily, weekly, ...
  - triggered: on “big” change, lots of changes, ...
- Data transformation
  - convert data to uniform format
  - remove & add fields (e.g., add date to get history)
- Standards (e.g., ODBC)
- Gateways
Integration

- Data Cleaning
- Data Loading
- Derived Data

**Data Cleaning**

- Migration (e.g., yen $\Rightarrow$ dollars)
- Scrubbing: use domain-specific knowledge (e.g., social security numbers)
- Fusion (e.g., mail list, customer merging)

- Auditing: discover rules & relationships (like data mining)
Loading Data

- Incremental vs. refresh
- Off-line vs. on-line
- Frequency of loading
  - At night, 1x a week/month, continuously
- Parallel/Partitioned load

Derived Data

- Derived Warehouse Data
  - indexes
  - aggregates
  - materialized views (next slide)
- When to update derived data?
- Incremental vs. refresh
Materialized Views

- Define new warehouse relations using SQL expressions

<table>
<thead>
<tr>
<th>sale</th>
<th>prodId</th>
<th>storeId</th>
<th>date</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>c1</td>
<td>1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>c1</td>
<td>1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c3</td>
<td>1</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>c2</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c1</td>
<td>2</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>c2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>product</th>
<th>id</th>
<th>name</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td></td>
<td>bolt</td>
<td>10</td>
</tr>
<tr>
<td>p2</td>
<td></td>
<td>nut</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>joinTb</th>
<th>prodId</th>
<th>name</th>
<th>price</th>
<th>storeId</th>
<th>date</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>bolt</td>
<td>10</td>
<td>c1</td>
<td>1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>nut</td>
<td>5</td>
<td>c1</td>
<td>1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>bolt</td>
<td>10</td>
<td>c3</td>
<td>1</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>nut</td>
<td>5</td>
<td>c2</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>bolt</td>
<td>10</td>
<td>c1</td>
<td>2</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>bolt</td>
<td>10</td>
<td>c2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

does not exist at any source

Processing

- ROLAP servers vs. MOLAP servers
- Index Structures
- What to Materialize?
- Algorithms
ROLAP Server

- Relational OLAP Server

  - tools
  - utilities
  - ROLAP server
  - relational DBMS

  Special indices, tuning:
  Schema is “denormalized”

<table>
<thead>
<tr>
<th>sale</th>
<th>prodId</th>
<th>date</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>1</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>1</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>2</td>
<td>48</td>
<td></td>
</tr>
</tbody>
</table>

MOLAP Server

- Multi-Dimensional OLAP Server

  - M.D. tools
  - utilities
  - multi-dimensional server

  could also sit on relational DBMS
Index Structures

- Traditional Access Methods
  - B-trees, hash tables, R-trees, grids, ...
- Popular in Warehouses
  - inverted lists
  - bit map indexes
  - join indexes
  - text indexes

Inverted Lists

![Inverted List Diagram](data/...)
Using Inverted Lists

- Query:
  - Get people with age = 20 and name = “fred”
- List for age = 20: r4, r18, r34, r35
- List for name = “fred”: r18, r52
- Answer is intersection: r18

Bit Maps

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>joe</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>fred</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>sally</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>nancy</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>tom</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>pat</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>dave</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>jeff</td>
<td>26</td>
</tr>
</tbody>
</table>

The bit maps for the age index are shown with 1s indicating presence and 0s indicating absence.
Using Bit Maps

- Query:
  - Get people with age = 20 and name = “fred”
  - List for age = 20: 1101100000
  - List for name = “fred”: 0100000001
  - Answer is intersection: 01000000000

- Good if domain cardinality small
- Bit vectors can be compressed

Join

- “Combine” SALE, PRODUCT relations
- In SQL: SELECT * FROM SALE, PRODUCT

### SALE table
<table>
<thead>
<tr>
<th>prodId</th>
<th>storeId</th>
<th>date</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>c1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>p2</td>
<td>c1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>p1</td>
<td>c3</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>p2</td>
<td>c2</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>p1</td>
<td>c1</td>
<td>2</td>
<td>44</td>
</tr>
<tr>
<td>p1</td>
<td>c2</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

### PRODUCT table
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>bolt</td>
<td>10</td>
</tr>
<tr>
<td>p2</td>
<td>nut</td>
<td>5</td>
</tr>
</tbody>
</table>

### JOIN table
<table>
<thead>
<tr>
<th>joinTb</th>
<th>prodId</th>
<th>name</th>
<th>price</th>
<th>storeId</th>
<th>date</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>bolt</td>
<td>10</td>
<td>c1</td>
<td>1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>nut</td>
<td>5</td>
<td>c1</td>
<td>1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>bolt</td>
<td>10</td>
<td>c3</td>
<td>1</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>nut</td>
<td>5</td>
<td>c2</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>bolt</td>
<td>10</td>
<td>c1</td>
<td>2</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>bolt</td>
<td>10</td>
<td>c2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
Join Indexes

Join index

<table>
<thead>
<tr>
<th>product</th>
<th>id</th>
<th>name</th>
<th>price</th>
<th>jIndex</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>bolt</td>
<td>10</td>
<td></td>
<td>r1,r3,r5,r6</td>
</tr>
<tr>
<td>p2</td>
<td>nut</td>
<td>5</td>
<td></td>
<td>r2,r4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sale</th>
<th>rId</th>
<th>prodId</th>
<th>storeId</th>
<th>date</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>p1</td>
<td>c1</td>
<td></td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>r2</td>
<td>p2</td>
<td>c1</td>
<td></td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>r3</td>
<td>p1</td>
<td>c3</td>
<td></td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>r4</td>
<td>p2</td>
<td>c2</td>
<td></td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>r5</td>
<td>p1</td>
<td>c1</td>
<td></td>
<td>2</td>
<td>44</td>
</tr>
<tr>
<td>r6</td>
<td>p1</td>
<td>c2</td>
<td></td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

What to Materialize?

- Store in warehouse results useful for common queries
- Example:

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

129
```

Materialize
Materialization Factors

- Type/frequency of queries
- Query response time
- Storage cost
- Update cost

Cube Aggregates Lattice

use greedy algorithm to decide what to materialize
Dimension Hierarchies

- all
  - state
    - city

<table>
<thead>
<tr>
<th>cities</th>
<th>city</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>CA</td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>NY</td>
<td></td>
</tr>
</tbody>
</table>

Dimension Hierarchies

- all
  - city
  - product
  - date
  - city, product
  - city, date
  - product, date
  - state
  - state, product
  - state, date
  - state, product, date

not all arcs shown...
Interesting Hierarchy

<table>
<thead>
<tr>
<th>time</th>
<th>day</th>
<th>week</th>
<th>month</th>
<th>quarter</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2000</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2000</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2000</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2000</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2000</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2000</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2000</td>
</tr>
</tbody>
</table>

conceptual dimension table

Design

- What data is needed?
- Where does it come from?
- How to clean data?
- How to represent in warehouse (schema)?
- What to summarize?
- What to materialize?
- What to index?
Tools

- Development
  - design & edit: schemas, views, scripts, rules, queries, reports

- Planning & Analysis
  - what-if scenarios (schema changes, refresh rates), capacity planning

- Warehouse Management
  - performance monitoring, usage patterns, exception reporting

- System & Network Management
  - measure traffic (sources, warehouse, clients)

- Workflow Management
  - “reliable scripts” for cleaning & analyzing data

Current State of Industry

- Extraction and integration done off-line
  - Usually in large, time-consuming, batches

- Everything copied at warehouse
  - Not selective about what is stored
  - Query benefit vs storage & update cost

- Query optimization aimed at OLTP
  - High throughput instead of fast response
  - Process whole query before displaying anything