The Need for Data Warehouse

- Decision support applications, e.g.
  - Show the sales number by month, by day, region, and/or by product
- Reporting and analysis applications, e.g.
  - Web site analytics
  - Online ad tracking

... The Need for Data Warehouse

- These applications are dominated by queries involving aggregations and group-bys
- And such queries often can't be expressed or executed efficiently by OLTP databases

Standard SQL Aggregation Functions

- Operate on multiple rows and return a single result
  - sum
  - avg
  - count
  - max and min

More About Aggregation Functions

- Distributive
  - sum, count, min, max
- Algebraic
  - avg = sum / count
- Holistic
  - median
Distributive Aggregation

<table>
<thead>
<tr>
<th>Count</th>
<th>Sum</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>[5,6,2,8,1,9]</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[11,12,14,16,18]</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[23,20]</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>??</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Holistic Aggregation

<table>
<thead>
<tr>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>[5,6,2,8,1,9]</td>
</tr>
<tr>
<td>[11,12,14,16,18]</td>
</tr>
<tr>
<td>[23,20]</td>
</tr>
<tr>
<td>All</td>
</tr>
</tbody>
</table>

Estimate Median ...

<table>
<thead>
<tr>
<th>Count</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>[5,6,2,8,1,9]</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>[11,12,14,16,18]</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>[23,20]</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>All</td>
<td>13</td>
<td>1</td>
</tr>
</tbody>
</table>

... Estimate Median

\[
\text{median} = \text{Min}_m + \frac{N/2 - \sum \text{Count}_n}{\text{Count}_n} \cdot (\text{Max}_n - \text{Min}_n)
\]

N: total count
m: the median interval
l: the intervals lower than the median interval

GROUP BY

```
SELECT category, COUNT(id)
FROM products
GROUP BY category;
```

Understanding GROUP BY ...

Without aggregation/GROUP BY ...

```
SELECT category, id FROM products;
```

<table>
<thead>
<tr>
<th>id</th>
<th>category</th>
<th>description</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CPU</td>
<td>Intel Core 2 Duo</td>
<td>$200.99</td>
</tr>
<tr>
<td>2</td>
<td>CPU</td>
<td>Intel Pentium D</td>
<td>$98.99</td>
</tr>
<tr>
<td>3</td>
<td>CPU</td>
<td>AMD Athlon 64</td>
<td>$74.99</td>
</tr>
<tr>
<td>4</td>
<td>CPU</td>
<td>AMD Athlon 64x2</td>
<td>$115.98</td>
</tr>
<tr>
<td>5</td>
<td>HD</td>
<td>Seagate 320G</td>
<td>$77.49</td>
</tr>
<tr>
<td>6</td>
<td>HD</td>
<td>Maxtor 250G</td>
<td>$60.89</td>
</tr>
</tbody>
</table>
... Understanding GROUP BY

- With aggregation/GROUP BY

```
select category, count(id) from products group by category;
```

<table>
<thead>
<tr>
<th>category</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1</td>
</tr>
<tr>
<td>CPU</td>
<td>2</td>
</tr>
<tr>
<td>CPU</td>
<td>3</td>
</tr>
<tr>
<td>CPU</td>
<td>4</td>
</tr>
<tr>
<td>HD</td>
<td>5</td>
</tr>
<tr>
<td>HD</td>
<td>6</td>
</tr>
</tbody>
</table>

- Grouping attribute
- Aggregation attribute

count(id) = 4

- count(id) = 2

Data Warehouse

- “A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management’s decision making process” – W. H. Inmon
- An Online Analytical Processing (OLAP) system

Data Warehouse Architecture

- Query/report
- Analysis
- Data mining

- OLAP servers
- Monitoring
- Admin
- Data warehouse servers
- Metadata repository
- Operation databases
- External data sources

Data Warehouse vs. Operational Database

<table>
<thead>
<tr>
<th></th>
<th>Operational Database</th>
<th>Data Warehouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>Detailed and current</td>
<td>??</td>
</tr>
<tr>
<td>Users</td>
<td>Clients, developers, DBA</td>
<td>??</td>
</tr>
<tr>
<td>Access Patterns</td>
<td>short, atomic, r/w transactions</td>
<td>??</td>
</tr>
<tr>
<td>Design</td>
<td>ER, normalized</td>
<td>??</td>
</tr>
</tbody>
</table>

Data

- Customer John Doe, whose address is 123 Main St., LA, CA, bought an Intel CPU for $279 and two Seagate hard drives for $300 at the Best Buy store on Foothill Blvd on 1/9/2012 at 11:01am.
Why Not Use Operational Database for OLAP

- Detailed, normalized data is not suitable for efficient OLAP operations.
- ER/relational model is good for data storage and access but not for data analysis.

The Multidimensional Model

Terminology

- Dimensions
  - Time, product, location ...
- Facts
  - Sales, units sold, expenses ...

Star Schema ...

... Star Schema

- One Fact Table
  - E.g. sales
- One Dimension Table per dimension
  - E.g. time, product, and store

From Operational Database to Star Schema ...

- Fact table
  - Data selection
- Data granularity (i.e. base facts)
- Derived data
- Pre-aggregated data (i.e. summary facts)
From Operational Database to Star Schema

- Dimension tables
  - Dimension selection
  - Time dimension
  - De-normalization
  - Surrogate key and natural key

Other Schemas for Multidimensional Databases

- Snowflake schema
  - Some dimensions are normalized
- Fact Constellation schema
  - Dimension tables are shared by more than one fact tables

Concept Hierarchies

```
country
  ↑
state
  ↑
city
  ↑
street

year
  ↑
quarter
  ↑
month
  ↑
week
  ↑
day
```

- **Total order:** street < city < state < country
- **Partial order:** day < (month < quarter, week) < year

OLAP Operations

- Roll-up
- Drill-down
- Slice and dice
- Pivot (rotate)

Roll-up

- Aggregation by
  - Going up a concept hierarchy, or
  - Reducing dimension(s)

```
  group by month, city
  group by year, city
  group by month
```

Drill-down

- Reverse of roll-up
  - Going down a concept hierarchy, or
  - Adding dimensions
Slice and Dice

- **Slice**: selection on one dimension
- **Dice**: selection on more than one dimensions
  - E.g. `(city='LA')` and `(month='Jan' or month='March')`

Pivot (Rotate)

- Rotate the data axes to provide an alternative presentation of the data

Perform OLAP Operations Efficiently

- **Indexing**
- **Pre-computation**
  - Summary fact tables
  - **Data cubes**

Bitmap Indexing ...

<table>
<thead>
<tr>
<th>rid</th>
<th>item</th>
<th>city</th>
<th>month</th>
<th>sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>TV</td>
<td>LA</td>
<td>Jan</td>
<td>100</td>
</tr>
<tr>
<td>1002</td>
<td>PC</td>
<td>LA</td>
<td>Jan</td>
<td>200</td>
</tr>
<tr>
<td>1003</td>
<td>PC</td>
<td>NY</td>
<td>Jan</td>
<td>150</td>
</tr>
<tr>
<td>1004</td>
<td>PC</td>
<td>NY</td>
<td>Feb</td>
<td>100</td>
</tr>
<tr>
<td>1005</td>
<td>Phone</td>
<td>NY</td>
<td>Jan</td>
<td>175</td>
</tr>
<tr>
<td>1006</td>
<td>TV</td>
<td>NY</td>
<td>Feb</td>
<td>200</td>
</tr>
<tr>
<td>1007</td>
<td>Phone</td>
<td>LA</td>
<td>Jan</td>
<td>300</td>
</tr>
<tr>
<td>1008</td>
<td>Phone</td>
<td>LA</td>
<td>Feb</td>
<td>120</td>
</tr>
</tbody>
</table>

... Bitmap Indexing

Bitmap Index on Item:  

```
1 0 0
0 1 0
0 1 0
0 1 0
0 0 1
1 0 0
0 0 1
0 0 1
```

TV  PC  Phone

Bitmap Index on City ??

Using Bitmap Index

- List total sales in LA by item

```sql
select sum(sales), item
from sales_table
where city = 'LA'
group by item;
```
Join Indexing ...

location
  | Location_key | Street | City | State | Country |
  | Item_key     | Item_name | Type |
item
  | Item_key | Item_name | Brand |

... Join Indexing ...

<table>
<thead>
<tr>
<th>location</th>
<th>Street</th>
<th>City</th>
<th>State</th>
<th>Country</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location_key</td>
<td>Item_key</td>
<td>Loc_key</td>
<td>Amount</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sales &amp; Item type</th>
<th>Sales &amp; Item type &amp; City</th>
</tr>
</thead>
<tbody>
<tr>
<td>item_type</td>
<td>city</td>
</tr>
<tr>
<td>1001</td>
<td>TV</td>
</tr>
<tr>
<td>1003</td>
<td>PC</td>
</tr>
<tr>
<td>1005</td>
<td>Phone</td>
</tr>
<tr>
<td>1007</td>
<td>Phone</td>
</tr>
</tbody>
</table>

Using Join Index

Find the total sales of TV
Find the total sales of TV in LA

Readings

Textbook Chapter 4