Main contents of today:

- Data Warehouse (and Business Intelligence)
- Data Cube abstraction (Pivot Table, OLAP)
- Multi-dimensional modeling (Star, Snowflake)
- **Advanced**: Complex and evolving dimensions...

Acknowledgment

- This slide deck is a “mashup” of the following publicly available slide decks:
  - http://www.postech.ac.kr/~swhwang/grass/DataCube.ppt
  - http://www.cs.uiuc.edu/homes/hanj/bk2/03.ppt
  - Hector Garcia-Molina, Stanford University
  - Marlon Dumas, Univ. of Tartu,
  - Sulev Reisberg, Quretec & STACC
  - Torben Bach Pedersen , Aalborg University, DK
  - ...

What is Business Intelligence (BI)?

- From Encyclopedia of Database Systems:
  "[BI] refers to a set of tools and techniques that enable a company to transform its business data into timely and accurate information for the decisional process, to be made available to the right persons in the most suitable form."

What is Business Intelligence (BI)?

- BI is different from Artificial Intelligence (AI)
  - AI systems make decisions for the users
  - BI systems help the users make the right decisions, based on available data
- Combination of technologies
  - Data Warehousing (DW)
  - On-Line Analytical Processing (OLAP)
  - Data Mining (DM)
  - …

Case Study of an Enterprise

- Example of a chain (e.g., fashion stores or car dealers)
  - Each store maintains its own customer records and sales records
  - Hard to answer questions like: “find the total sales of Product X from stores in Aalborg”
  - The same customer may be viewed as different customers for different stores; hard to detect duplicate customer information
  - Imprecise or missing data in the addresses of some customers
  - Purchase records maintained in the operational system for limited time (e.g., 6 months); then they are deleted or archived
  - The same “product” may have different prices, or different discounts in different stores
- Can you see the problems of using those data for business analysis?
Data Analysis Problems

- The same data found in many different systems
  - Example: customer data across different stores and departments
  - The same concept is defined differently

- Heterogeneous sources
  - Relational DBMS, On-Line Transaction Processing (OLTP)
  - Unstructured data in files (e.g., MS Word)
  - Legacy systems
  - …

Data Analysis Problems (cont')

- Data is suited for operational systems
  - Accounting, billing, etc.
  - Does not support analysis across business functions

- Data quality is bad
  - Missing data, imprecise data, different use of systems

- Data is “volatile”
  - Data deleted in operational systems (6 months)
  - Data changes over time – no historical information

- Kimball & Ross point out typical issues:
  - “We have mountains of data, but we can’t access it”
  - “We need to slice and dice the data in every which way”
  - “Make it easy to get the data directly”
  - “Show me what is important”
  - “Two people present the business metrics, but with different numbers”

- It is time for a change …

Data Warehousing

- Solution: new analysis environment (DW) where the data is
  - Subject oriented (versus function oriented)
  - Integrated (logically and physically)
  - Time variant (data can always be related to time)
  - Stable (data not deleted, several versions)
  - Supporting management decisions (different organization)

- Data from the operational systems is
  - Extracted
  - Cleansed
  - Transformed
  - Aggregated (?)
  - Loaded into the DW

- A good DW is a prerequisite for successful BI

What is a Data Warehouse?

- A DW is a store of information organized in a unified data model
- Data collected from a number of different sources
  - Finance, billing, website logs, personnel, …
- Purpose of a data warehouse (DW):
  - support decision making
- Easy to perform advanced analysis
  - Ad-hoc analysis and reports
  - We will cover this soon …
- Data mining: discovery of hidden patterns and trends

Figure 2: The data warehouse has a strong subject orientation.
Hard/Infeasible Queries for OLTP

- Why not use the existing databases (OLTP) for business analysis?
  - Business analysis queries
    - In the past five years, which 10 products are most profitable?
    - Which public holiday has the largest sales?
    - Which week has the largest sales?
    - Does the sales of dairy products increase over time?
  - Difficult to express these queries in SQL
    - 3rd query: we may extract the “week” value using a function
      - But the user has to learn many transformation functions ...
    - 4th query: use a “special” table to store IDs of all dairy products, in advance
      - There can be many different dairy products; there can be many other product types as well ...
    - There is a need for multidimensional modeling ...

OLTP vs. OLAP

- OLTP – Online Transaction Processing
  - Traditional database technology
  - Many small transactions (point queries: UPDATE or INSERT)
  - Avoid redundancy, normalize schemas
  - Access to consistent, up-to-date database
- OLTP Examples:
  - Flight reservation
  - Banking and financial transactions
  - Order Management, Procurement, ...
- Extremely fast response times...

OLTP vs. OLAP (Water and Oil)

- Lock Conflicts: OLAP blocks OLTP
- Database design:
  - OLTP normalized, OLAP de-normalized
- Tuning, Optimization
  - OLTP: inter-query parallelism, heuristic optimization
  - OLAP: intra-query parallelism, full-fledged optimization
- Freshness of Data:
  - OLTP: serializability
  - OLAP: reproducibility
  - OLAP: Sampling, Confidence Intervals
- Integrity:
  - OLTP: ACID
  - OLAP: Sampling, Confidence Intervals

OLTP vs. OLAP

- OLAP – Online Analytical Processing
  - Big aggregate queries, no Updates
  - Redundancy a necessity (Materialized Views, special-purpose indexes, de-normalized schemas)
  - Periodic refresh of data (daily or weekly)
- OLAP Examples
  - Decision support (sales per employee)
  - Marketing (purchases per customer)
  - Biomedical databases
- Goal: Response Time of seconds / few minutes
Solution: Data Warehouse

- Special Sandbox for OLAP
- Data input using OLTP systems
- Data Warehouse aggregates and replicates data (special schema)
- New Data is periodically uploaded to Warehouse

2. Data Cube abstraction

- Pivot Table
- The “data cube” abstraction
- Multidimensional data models

Example tool: TARGIT BI Suite

Example: Sales

Excel pivot table

Sales data example
Multidimensional View of Sales

- Multidimensional analysis involves viewing data simultaneously categorized along potentially many dimensions

Pivoting

Typical Data Analysis Process

- Formulate a query to extract relevant information
- Extract aggregated data from the database
- Visualize the result to look for patterns.
- Analyze the result and formulate new queries.
- Online Analytical Processing (OLAP) is about supporting such processes
- OLAP characteristics: No updates, lots of aggregation, need to visualize and to interact
- Let’s first talk about aggregation...

Relational Aggregation Operators

- SQL has several aggregate operators:
  - SUM(), MIN(), MAX(), COUNT(), AVG()
- The basic idea is:
  - Combine all values in a column into a single scalar value
- Syntax
  - SELECT AVG(Temp) FROM Weather;
Limitations of the GROUP BY

• Group-by is one-dimensional: one group per combination of the selected attribute values

   
<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>

1. Calculate total sales per year
2. Compute total sales per year and per color
3. Calculate sales per year, per color and per model

Grouping with Sub-Totals (Pivot table)

• Sales by Model by Year by Color

   
   ![](image1)

   
   Notes that sub-totals by color are missing, if added it becomes a cross-tabulation

Grouping with sub-totals (cross-tab)

Sub-totals by color are still missing...

SQL Query

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

Adding the colors...

```
SELECT 'ALL', 'ALL', 'ALL', SUM(Sales)
FROM Sales
GROUP BY Model = 'Chevy'
UNION
SELECT Model, ALL, ALL, SUM(Sales)
FROM Sales
GROUP BY Model = 'Chevy'
UNION
SELECT Model, Year, ALL, SUM(Sales)
FROM Sales
GROUP BY Model, Year
UNION
SELECT Model, Year, Color, SUM(Sales)
FROM Sales
GROUP BY Model, Year, Color;
```
CUBE and Roll Up Operators

The Cube
• An Example of 3D Data Cube

CUBE Operator
Possible syntax

- Proposed syntax example:
  ```sql
  SELECT Model, Make, Year, SUM(Sales)
  FROM Sales
  WHERE Model IN {"Chevy", "Ford"}
  AND Year BETWEEN 1990 AND 1994
  GROUP BY CUBE Model, Make, Year
  HAVING SUM(Sales) > 0;
  ```
- Note: GROUP BY operator repeats aggregate list
  • in select list
  • in group by list
### Cube Operator Example

<table>
<thead>
<tr>
<th>Model</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>87</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>

### Summary

- **Problems with GROUP BY**
  - GROUP BY cannot directly construct
    - Pivot tables / roll-up reports
    - Cross-Tabs
- **CUBE Operator**
  - Generalizes GROUP BY and Roll-Up and Cross-Tabs!!

---

### Data Cubing

- **Slicing**
- **Dicing**
- **Drill-up and Drill-Down**
- **Pivoting**
Roll-up: A roll-up involves summarizing the data along a dimension. The summarization rule might be computing totals along a hierarchy or applying a set of formulas such as "profit = sales - expenses".\[^{[5]}\]

Rollup Operator

ROLLUP Operator: special case of CUBE Operator

Return "Sales Roll Up by Store by Quarter" in 1994:

```
SELECT Store, quarter, SUM(Sales)
FROM Sales
WHERE nation="Korea" AND Year=1994
GROUP BY ROLLUP Store, Quarter(Date) AS quarter;
```
3. Multidimensional modelling

- Sales volume as a function of product, month, and region

**Multidimensional Data**

- Dimensions: Product, Location, Time
- Hierarchical summarization paths

**Star Schema**

**Dimension Hierarchies**

- Fact table
- Dimension tables
- Measures

- 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>12</td>
</tr>
<tr>
<td>p2</td>
<td>c1</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>p1</td>
<td>c3</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>p2</td>
<td>c2</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Multi-dimensional cube:

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>12</td>
</tr>
<tr>
<td>p2</td>
<td>c1</td>
<td>1</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>p1</td>
<td>c3</td>
<td>1</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>p2</td>
<td>c2</td>
<td>1</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Multi-dimensional cube:

dimensions = 3

Star Schema

<table>
<thead>
<tr>
<th>time_key</th>
<th>day</th>
<th>day_of_the_week</th>
<th>month</th>
<th>quarter</th>
<th>year</th>
<th>time</th>
<th>location_key</th>
<th>location</th>
<th>street</th>
<th>city_key</th>
<th>city</th>
<th>state_or_province</th>
<th>country</th>
<th>item_key</th>
<th>item_name</th>
<th>brand</th>
<th>type</th>
<th>supplier_type</th>
</tr>
</thead>
</table>

| branch_key | branch_name | branch_type | location_key | location | street | city_key | city | state_or_province | country | item_key | item_name | brand | type | supplier_type |

| units_sold | dollars_sold | avg_sales |             |         |        |        |      |                |         |         |           |       |      |               |

Snowflake Schema

<table>
<thead>
<tr>
<th>time_key</th>
<th>day</th>
<th>day_of_the_week</th>
<th>month</th>
<th>quarter</th>
<th>year</th>
<th>time</th>
<th>location_key</th>
<th>location</th>
<th>street</th>
<th>city_key</th>
<th>city</th>
<th>state_or_province</th>
<th>country</th>
<th>item_key</th>
<th>item_name</th>
<th>brand</th>
<th>type</th>
<th>supplier_type</th>
</tr>
</thead>
</table>

| branch_key | branch_name | branch_type | location_key | location | street | city_key | city | state_or_province | country | item_key | item_name | brand | type | supplier_type |

| units_sold | dollars_sold | avg_sales |             |         |        |        |      |                |         |         |           |       |      |               |

Case study: Hospital

What is data warehouse

- Information system for reporting purposes
- The goal is to fulfill reporting needs which are unsatisfied in operational system
  - It is easy to modify old and design new reports
    - No „write spec to software developer to get the report“ anymore
    - Reports can be filled with data quickly
    - No „start the report generation at night to prevent system load“ anymore
  - The data comes from operational system(s)

Goal of the work package

- Work out the main concepts for building data warehouse for hospital IS
  - What are the reporting needs?
  - What are the data cubes that cover most reporting needs for „universal“ hospital?
  - How to get the data into these cubes?
Partners in this work package

- **Ida-Tallinna Keskkelaigla (ITK)**
  - One of the biggest hospitals in Estonia
  - Huge amount of data in operational system (system called ESTER)
  - Has difficulties in generating reports on operational system
  - Interested in improving the report management

- **Quretec**
  - Provides data management software for different clients in Europe, especially in healthcare area
  - Interested in increasing the knowledge of data warehousing area

So far... (1)
- We have analyzed the data and data structures in operational system

So far... (2)
- We have designed the interface for getting the data from ESTER
- We have built 2 data cubes

So far... (3)
- We have designed 10 reports on the data

So far... (4)
- Showed that report generation time **has reduced from tens of minutes to few seconds**

<table>
<thead>
<tr>
<th>Selected period</th>
<th>Number of patients</th>
<th>Seconds for generating report in operational system</th>
<th>Seconds for generating the same report in data warehouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 day</td>
<td>138</td>
<td>149</td>
<td>1</td>
</tr>
<tr>
<td>1 month</td>
<td>2944</td>
<td>150</td>
<td>1</td>
</tr>
<tr>
<td>1 year</td>
<td>32286</td>
<td>158</td>
<td>1</td>
</tr>
</tbody>
</table>

So far... (5)
- We showed that data warehouse offers additional benefits:
  - Multiple output formats
  - Reports can be redesigned easily
  - **New combined reports** -> new value from the data
Implementing a Warehouse

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

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

Incremental vs. Refresh

<table>
<thead>
<tr>
<th>customer id</th>
<th>name</th>
<th>address</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
<td>Joe</td>
<td>10 main</td>
<td>sfo</td>
</tr>
<tr>
<td>81</td>
<td>Fred</td>
<td>12 main</td>
<td>sfo</td>
</tr>
<tr>
<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)
  - billing DB \(\rightarrow\) customer1(Joe) \(\rightarrow\) merged_customer(Joe)
  - service DB \(\rightarrow\) customer2(Joe)
- Auditing: discover rules & relationships (like data mining)
Loading Data

- Incremental vs. refresh
- Offline 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

ETL

- Export
- Transform
- Load

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?

ROLAP Server

- Relational OLAP Server

- Multi-Dimensional OLAP Server

<table>
<thead>
<tr>
<th>Product</th>
<th>City</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>milk</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>soda</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>eggs</td>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>soap</td>
<td>B</td>
<td>4</td>
</tr>
</tbody>
</table>

Special indices, tuning; Schema is "denormalized"

MOLAP Server

Multi-Dimensional OLAP Server

Utilities

Multi-dimensional server
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

- 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

Join

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

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

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 Indexes

<table>
<thead>
<tr>
<th>product</th>
<th>id</th>
<th>name</th>
<th>price</th>
<th>index</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>10</td>
<td>bolt</td>
<td>10</td>
<td>r1,r3,r5,r6</td>
</tr>
<tr>
<td>p2</td>
<td>20</td>
<td>nut</td>
<td>5</td>
<td>r2,r4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sale</th>
<th>tid</th>
<th>product</th>
<th>storeid</th>
<th>date</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>p1</td>
<td>c1</td>
<td>1</td>
<td>12</td>
<td>44</td>
</tr>
<tr>
<td>r2</td>
<td>p2</td>
<td>c1</td>
<td>1</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>r3</td>
<td>p1</td>
<td>c3</td>
<td>1</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>r4</td>
<td>p2</td>
<td>c2</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>r5</td>
<td>p1</td>
<td>c1</td>
<td>2</td>
<td>44</td>
<td>129</td>
</tr>
<tr>
<td>r6</td>
<td>p1</td>
<td>c2</td>
<td>2</td>
<td>4</td>
<td>129</td>
</tr>
</tbody>
</table>

What to Materialize?

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

```

<table>
<thead>
<tr>
<th>day 2</th>
<th>day 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>c1</td>
</tr>
<tr>
<td>c2</td>
<td>c2</td>
</tr>
<tr>
<td>c3</td>
<td>c3</td>
</tr>
<tr>
<td>p1</td>
<td>p1</td>
</tr>
<tr>
<td>44</td>
<td>56</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>129</td>
<td>129</td>
</tr>
</tbody>
</table>
```

Materialization Factors

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

Cube Aggregates Lattice

Interesting Hierarchy

Changing Dimensions

- So far, we assumed that dimensions are stable over time
  - New rows in dimension tables can be inserted
  - Existing rows do not change
    - This is not a realistic assumption
- We now study techniques for handling changes in dimensions
  - “Slowly changing dimensions” phenomenon
    - Dimension information change, but changes are not frequent
    - Still assume that the schema is fixed

Use greedy algorithm to decide what to materialize
Advanced: Handling Changes in Dimensions

- Handling change over time
- Changes in dimensions
  - 1. No special handling
  - 2. Versioning dimension values
    - Versioning dimension values
  - 3. Capturing the previous and the current value
  - 4. Split into changing and constant attributes

Example

Store dim.
- Attribute values in dimensions vary over time
- A store changes Size
- A product changes Description
- Districts are changed
- Problems
  - Dimensions not updated
  - A store is not up-to-date
  - Dimensions updated in a straightforward way
  - Incorrect information in historical data

Solution 1: No Special Handling

Sales fact table
<table>
<thead>
<tr>
<th>StoreID</th>
<th>ItemsSold</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>2000</td>
<td></td>
</tr>
</tbody>
</table>

Store dimension table
<table>
<thead>
<tr>
<th>StoreID</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>250</td>
</tr>
</tbody>
</table>

Solution 2

Solution 2: Versioning of rows with changing attributes

- The key that links dimension and fact table, identifies a version of a row, not just a "row"
- Surrogate keys make this easier to implement
  - What if we had used, e.g., the shop’s zip code as key?
  - Always use surrogate keys!!

- Consequences
  - Larger dimension tables
  - Pros
    - Correct information captured in DW
    - No problems when formulating queries
  - Cons
    - Cannot capture the development over time of the subjects the dimensions describe in the simplest form (but we can fix that)
Solution 2: Versioning of Rows

<table>
<thead>
<tr>
<th>StoreID</th>
<th>ItemsSold</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>2000</td>
</tr>
<tr>
<td>001</td>
<td>250</td>
</tr>
<tr>
<td>002</td>
<td>3500</td>
</tr>
</tbody>
</table>

A new fact arrives

<table>
<thead>
<tr>
<th>StoreID</th>
<th>ItemsSold</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>2000</td>
</tr>
<tr>
<td>002</td>
<td>3500</td>
</tr>
</tbody>
</table>

Which store does the new fact (old fact) refer to?

Solution 2A

Solution 2A: Inserting Special Facts

<table>
<thead>
<tr>
<th>StoreID</th>
<th>TimeID</th>
<th>ItemsSold</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>234</td>
<td>2000</td>
</tr>
<tr>
<td>002</td>
<td>345</td>
<td>-</td>
</tr>
<tr>
<td>002</td>
<td>456</td>
<td>3500</td>
</tr>
</tbody>
</table>

Solution 2B

Solution 2B: Timestamping

<table>
<thead>
<tr>
<th>StoreID</th>
<th>TimeID</th>
<th>ItemsSold</th>
<th>Size</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>234</td>
<td>2000</td>
<td>250</td>
<td>1998</td>
<td>-</td>
</tr>
<tr>
<td>001</td>
<td>234</td>
<td>2000</td>
<td>250</td>
<td>1998</td>
<td>1999</td>
</tr>
<tr>
<td>002</td>
<td>456</td>
<td>3500</td>
<td>450</td>
<td>2000</td>
<td>-</td>
</tr>
</tbody>
</table>

Example of Using Solution 2B

<table>
<thead>
<tr>
<th>StoreID</th>
<th>TimeID</th>
<th>ItemsSold</th>
<th>Size</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>234</td>
<td>2000</td>
<td>250</td>
<td>1998</td>
<td>-</td>
</tr>
<tr>
<td>001</td>
<td>234</td>
<td>2000</td>
<td>250</td>
<td>1998</td>
<td>1999</td>
</tr>
<tr>
<td>002</td>
<td>456</td>
<td>3500</td>
<td>450</td>
<td>2000</td>
<td>-</td>
</tr>
</tbody>
</table>

Solution 2A

- **Solution 2A**: Use special facts for capturing changes in dimensions via the Time dimension
  - Assume that no simultaneous, new fact refers to the new dimension row
  - Insert a new special fact that points to the new dimension row, and through its reference to the Time dimension, timestamps the row
- **Pros**
  - Possible to capture the development over time of the subjects that the dimensions describe
- **Cons**
  - Larger fact table
  - Cumbersome to use special facts in queries

Solution 2B

- **Solution 2B**: Versioning of rows with changing attributes like in Solution 2 + timestamping of rows in the slowly changing dimension (SCD) with From and To attributes
- **Pros**
  - Correct information captured in DW
- **Cons**
  - Larger dimension tables

Example of Using Solution 2B

- Product descriptions are versioned, when products are changed, e.g., new package sizes
  - Old versions are still in the stores, new facts can refer to both the newest and older versions of products
  - Time value for a fact not necessarily between “From” and “To” values in the fact’s Product dimension row
- Unlike changes in Size for a store, where all facts from a certain point in time will refer to the newest Size value
- Unlike alternative categorizations that one wants to choose between
**Solution 3**

- **Solution 3**: Create two versions of each changing attribute
  - One attribute contains the current value
  - The other attribute contains the previous value
- **Consequences**
  - Two values are attached to each dimension row
- **Pros**
  - Possible to compare across the change in dimension value (which is a problem with Solution 2)
  - Such comparisons are interesting when we need to work simultaneously with two alternative values
  - Example: Categorization of stores and products
- **Cons**
  - Not possible to see when the old value changed to the new
  - Only possible to capture the two latest values

**Rapidly Changing Dimensions**

- **Difference between “slowly” and “rapidly” is subjective**
  - Solution 2 is often still feasible
  - The problem is the size of the dimension
- **Example**
  - Assume an Employee dimension with 100,000 employees, each using 2K bytes and many changes every year
  - Solution 2 is recommended
  - Examples of (large) dimensions with many changes: Product and Customer
  - The more attributes in a dimension table, the more changes per row are expected
  - **Example**
    - A Customer dimension with 100M customers and many attributes
    - Solution 2 yields a dimension that is too large

**Solution 4**

- **Solution 4**: Dimension Splitting
  - Make a “minidimension” with the often-changing attributes
  - Convert (numeric) attributes with many possible values into attributes with few discrete or banded values
    - E.g., Income group: [0,10K), [0,20K), [0,30K), [0,40K)
      - Why? Any Information Loss?
  - Insert rows for all combinations of values from these new domains
    - With 6 attributes with 10 possible values each, the dimension gets $10^6 = 1,000,000$ rows
    - What do we do, if there are too many (theoretical) combinations?
    - If the minidimension is too large, it can be further split into more minidimensions
    - Here, synchronous/correlated attributes must be considered (to be placed in the same minidimension)
    - The same attribute can be repeated in another minidimension

**Solution 4 (Changing Dimensions)**

- **Pros**
  - DW size (dimension tables) is kept down
  - Changes in a customer’s profile values do not result in changes in dimensions
- **Cons**
  - More dimensions and more keys in the star schema
  - Navigation of customer attributes is more cumbersome as these are in more than one dimension
  - Using value groups gives less detail
  - The construction of groups is irreversible
Changing Dimensions - Summary

• Why are there changes in dimensions?
  • Applications change
  • The modeled reality changes
• Multidimensional models realized as star schemas support change over time to a large extent
• A number of techniques for handling change over time at the instance level was described
  • Solution 2 and the derived 2B are the most useful
  • Possible to capture change precisely

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

DW Products and Tools (old)

• Oracle 11g, IBM DB2, Microsoft SQL Server, ...
  – All provide OLAP extensions
• SAP Business Information Warehouse
  – ERP vendors
• MicroStrategy, Cognos (now IBM)
  – Specialized vendors
  – Kind of Web-based EXCEL
• Niche Players (e.g., Btell)
  – Vertical application domain

Chapter 2: Data Preprocessing

• Why preprocess the data?
• Data cleaning
• Data integration and transformation
• Data reduction
• Discretization and concept hierarchy generation
• Summary

Discretization

• Three types of attributes:
  • Nominal — values from an unordered set, e.g., color, profession
  • Ordinal — values from an ordered set, e.g., military or academic rank
  • Continuous — real numbers, e.g., integer or real numbers
• Discretization:
  • Divide the range of a continuous attribute into intervals
  • Some classification algorithms only accept categorical attributes.
  • Reduce data size by discretization
  • Prepare for further analysis
Discretization and Concept Hierarchy

- **Discretization**
  - Reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals
  - Interval labels can then be used to replace actual data values
  - Supervised vs. unsupervised
  - Split (top-down) vs. merge (bottom-up)
  - Discretization can be performed recursively on an attribute

- **Concept hierarchy formation**
  - Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for age) by higher level concepts (such as young, middle-aged, or senior)

Segmentation by Natural Partitioning

- A simply 3-4-5 rule can be used to segment numeric data into relatively uniform, “natural” intervals.
  - If an interval covers 3, 6, 7 or 9 distinct values at the most significant digit, partition the range into 3 equi-width intervals
  - If it covers 2, 4, or 8 distinct values at the most significant digit, partition the range into 4 intervals
  - If it covers 1, 5, or 10 distinct values at the most significant digit, partition the range into 5 intervals

Example of 3-4-5 Rule

<table>
<thead>
<tr>
<th>Range</th>
<th>Number of Distinct Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-$1,000,000 .. 0]</td>
<td>1</td>
</tr>
<tr>
<td>(0 .. 1,000,000]</td>
<td>2</td>
</tr>
<tr>
<td>(1,000,000 .. 2,000,000]</td>
<td>3</td>
</tr>
<tr>
<td>(2,000,000 .. 5,000,000]</td>
<td>4</td>
</tr>
</tbody>
</table>

Example

- MIN=-351,976.00
- MAX=4,700,896.50
- LOW = 5th percentile -159,876
- HIGH = 95th percentile 1,838,761
- msd = 1,000,000 (most significant digit)
- LOW = -1,000,000 (round down) HIGH = 2,000,000 (round up)

Concept Hierarchy Generation for Categorical Data

- Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts
  - street < city < state < country
- Specification of a hierarchy for a set of values by explicit data grouping
  - {Urbana, Champaign, Chicago} < Illinois
- Specification of only a partial set of attributes
  - E.g., only street < city, not others
- Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values
  - E.g., for a set of attributes: {street, city, state, country}
Automatic Concept Hierarchy Generation

- Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set.
- The attribute with the most distinct values is placed at the lowest level of the hierarchy.
- Exceptions, e.g., weekday, month, quarter, year, country, province or state, city, street.

15 distinct values
365 distinct values
3567 distinct values
674,339 distinct values

Summary

- OLAP and DW – a way to summarise data
- Prepare data for further data mining and visualisation
- Fact table, aggregation, queries&indices, ...

Reference (highly recommended)

- http://citeseer.ist.psu.edu/old/392672.html
- Data Warehousing chapter of Jianwei Han’s textbook (chapter 3)
- http://www.hha.dk/ifi/BUSINESS_l/documents/What_is_a_Data_Warehouse.pdf

Homework

- Exercises 1 and 4 at:
- Multidimensional data modeling exercise in course Wiki pages