Data Warehouses

Chapter 12

OLTP vs OLAP

- *Operational Database:* a database designed to support the day-to-day transactions of an organization
- *Data Warehouse:* historical data is periodically trimmed from the operational database and moved to a database specifically designed for analysis
  - Term coined by Bill Inmon in early 1980s
  - Significant contributions by Ralph Kimball and others

OLTP vs OLAP

- Online Transaction Processing (OLTP):
  - High transaction volume
  - Each transaction uses relatively little data
  - Day-to-day activities; current data
- Online Analytical Processing (OLAP):
  - Relatively few transactions
  - Each transaction uses large amounts of data
  - Historical data; analysis and decision-making

Comparison of OLTP and OLAP Systems

OLTP Systems	OLAP Systems
Support day-t0-day operations	Support strategic analysis
Hold current data required for day-to-day transactions	Hold historic data required for trend analysis, including snapshots
Store transaction details	Store aggregated data with some details
Data is dynamic, changing with each transaction	Data is static, except for periodic additions
Queries are short-running and access relatively few rows of data	Queries are long-running and access many rows of data
High transaction volume	Medium to low transaction volume
Repetitive processing, usually with predictable usage pattern	Ad-hoc processing without a predictable usage pattern
Process oriented	Subject oriented
Large number of users	Relatively low number of users

Data Warehouses

- ***Data Warehouse***: A subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision-making process
  - Organized around major subjects of the enterprise
  - Integrated from multiple operational sources
  - Only accurate across a known time period
  - Not updated in real-time; new data added periodically (as often as needed)

Benefits of Data Warehousing

- Competitive Advantage – thanks to more timely and accurate data
- Better Decisions – Including accurate history in forecasts leads to better business projections
- High Return on Investment – Historical information helps organizations find the best ways to:
  - Reduce operating costs
  - Increase revenue

Challenges of Data Warehousing

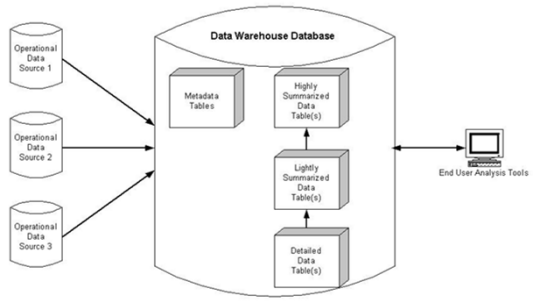
- Nailing Down Requirements
  - Countering the requirement to load everything
- Underestimation of Required Resources
  - DW projects typically not low cost
  - The never-ending project (often due to poor requirements)
- High Resource Demands
  - Huge storage requirements
  - Hundreds of millions of rows not unusual
- Hidden Data Integrity Problems
  - Source data is seldom perfect

Challenges of Data Warehousing (2)

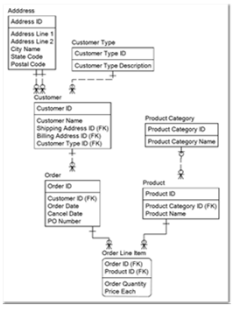
- Consolidating Data from Disparate Systems
  - Matching data from different sources can be difficult
  - Different keys and coding systems (e.g. In healthcare, HCPCS, CPT and ICD coding systems)
  - Late arriving data within some source systems
- Ownership of the Data
  - Owner not obvious once data is consolidated
  - Data Governance program often required
- Ever-Increasing End User Demands

DW Architecture: Summary Tables

Figure 12-1: Summary Table DW Architecture



Source System 3NF Structure

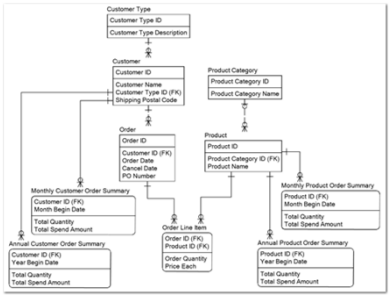


- DW Design Decisions
  - ZIP Codes useful, but rest of addresses are not
  - Order Line Item can be summarized by Product and Month, Product and Year, Customer and Month, and Customer and Year
  - Price Each cannot be summarized, multiply price by quantity and store extended amount

Summary Table Architecture

- Mostly normalized data structure
  - Summary tables added (to avoid repetitive summing)
- Key to success is finding the correct level of summarization
- Avoid keeping all the detail
  - Save that for an operational reporting database
  - Keeping some detail (e.g. large or unusual transactions) is o.k.

Summary Table Data Structure



Benefits of Inmon's Approach

- Schema is a closer match to the source system(s)
  - ETL is easier to write
  - Ongoing maintenance costs are lower (new tables usually fit easily with existing schema)
- Fits best when scope is the entire enterprise
  - Divides data by subject area, so more data centric
  - Star schema alternative is more process centric
- For tracking history, Inmon's approach allows for continuous time frames
  - Each row is typically devoted to a specific slice of time

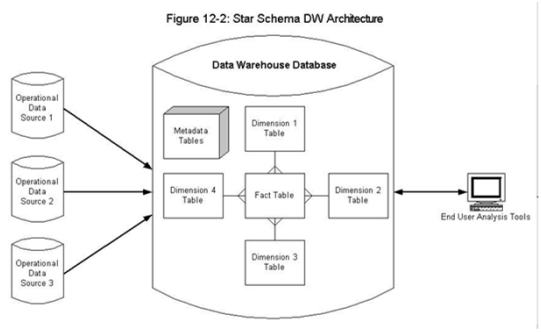
Drawbacks of Inmon's Approach

- Initial development takes longer
  - Sometimes much longer
- Star Schema is a better fit if the data being tracked are process measurements
  - Star Schema is more process oriented

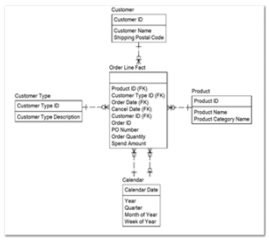
Star Schema Architecture

- Two types of tables:
  - ***Fact tables*** hold ***facts*** (measurements taken from business events)
  - **Dimension tables** hold ***attributes*** (data values that provide context)
    - Used for filtering, sorting, slicing (grouping for aggregation)
- Facts should be cumulative (business value not lost when summed)
- Fact tables usually do not have declared primary keys
- Dimensions establish the grain (level of detail) of the fact tables

Star Schema Architecture



Star Schema Data Structure



- No Summary tables
- No Hierarchies of Dimensions
  - Product Category consolidated into Product
  - Customer Type split and connected to fact table
- Calendar Date dimension added for all dates
- Spend Amount added
- Order attrs degenerated

Benefits of Kimball's Approach

- Faster initial development (compared to summary table)
- Better suited to process measurement facts
  - Star schemas are more process centric
- Star schemas are easier for business users to understand
  - Like pivot tables in Excel

Drawbacks of Kimball's Approach

- Schema is quite different than source system structure
  - ETL is more tedious to specify and develop
  - Ongoing maintenance costs are higher
- If scope is entire enterprise, summary table architecture may be a better choice
  - Summary table architecture is more data centric
- Does not handle continuous time frames as well as Inmon's approach
  - Handles dimension history with slowly changing dimensions

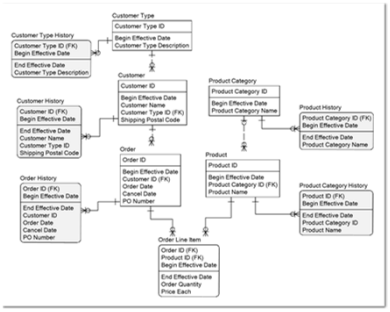
Handling History in Data Warehouses

- Applies mostly to dimension table attributes
- Three Classifications (from Kimball):
  - ***Type 1***: No history is kept
    - Data changes overwrite existing data
  - ***Type 2***: Full history of changes is kept
    - Data changes cause new dimension rows to be added
    - Dimension rows are effective dated or otherwise versioned
  - ***Type 3:*** Limited history is kept
    - Attributes (columns) added for previous data values
    - Seldom an enduring solution
  - At least 4 more types have been added (advanced forms)

Handling History in Summary Tables

- Usually two expensive to create new rows at top of hierarchy when something changes
  - Would require duplicating all the rows under the top row
- Typical Solution:
  - If table has children, add a history table under it, including effective dates
  - If table has no children, add effective dates (with begin date as part of the key)
  - When changes occur, old row must be "end-dated" when new row is added
  - Current row often has an end date with a known high date

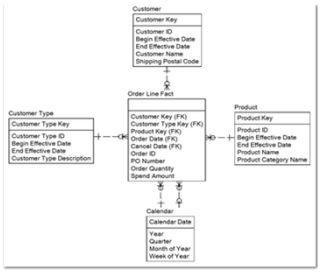
Summary Table with History



Handling History in Star Schemas

- Two Common Approaches:
  - Add Effective Dates to Type 2 Dimensions
    - A single attribute surrogate key is often used to simplify fact table foreign keys (and joins)
  - Snapshot Dimensions
    - Choose a time interval (e.g. daily, weekly, monthly)
    - Create a new dimension row for each time interval (even if nothing changed)
    - When joining facts to dimensions, logic can choose the point in time of interest (e.g. as of Booking Date vs. Shipping Date)

Star Schema with Effective Dated Dimensions



Star Schema with Snapshot Dimensions