Data Warehousing

Lecture-3
Dimensional Modeling (DM)

Dimensional Modeling (DM)

What is DM?

A simpler logical model optimized for decision support.

Inherently dimensional in nature, with a single central fact table and a set of smaller dimensional tables.

Multi-part key for the fact table

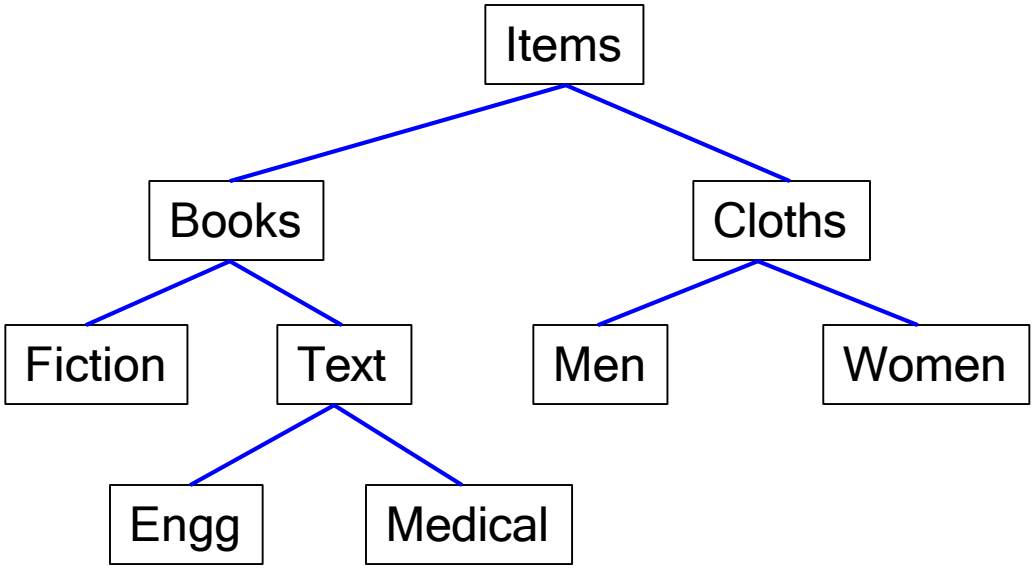
Dimensional tables with a single-part PK.

Keys are usually system generated

What is DM?

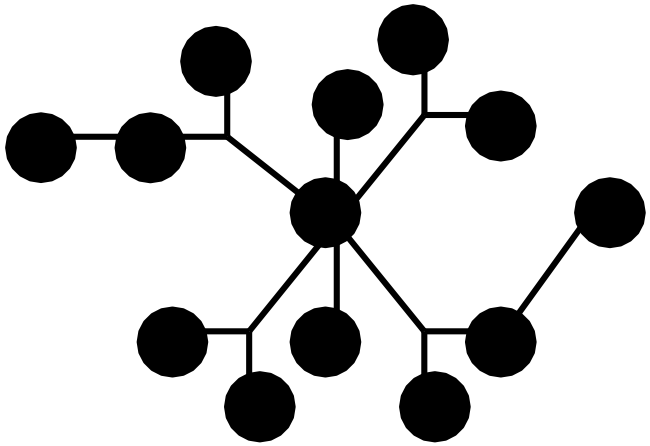
- Results in a star like structure, called star schema or star join.
  - All relationships mandatory M-1.
  - Single path between any two levels.
- Supports ROLAP operations.

Dimensions have Hierarchies

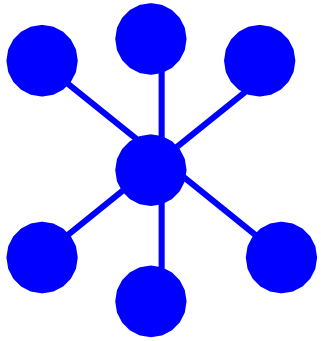


Analysts tend to look at the data through dimension at a particular “level” in the hierarchy

The two Schemas

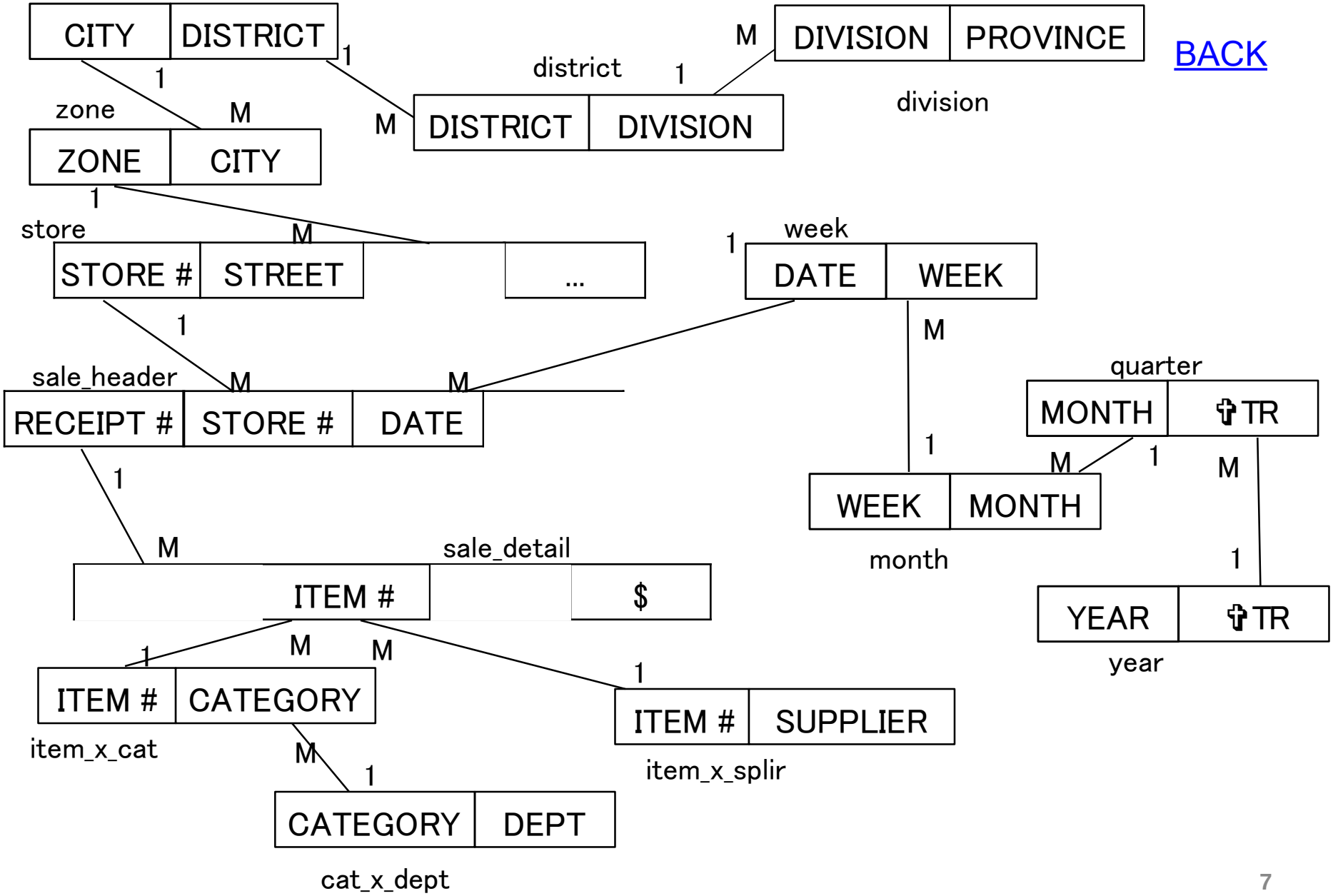


Snow-flake

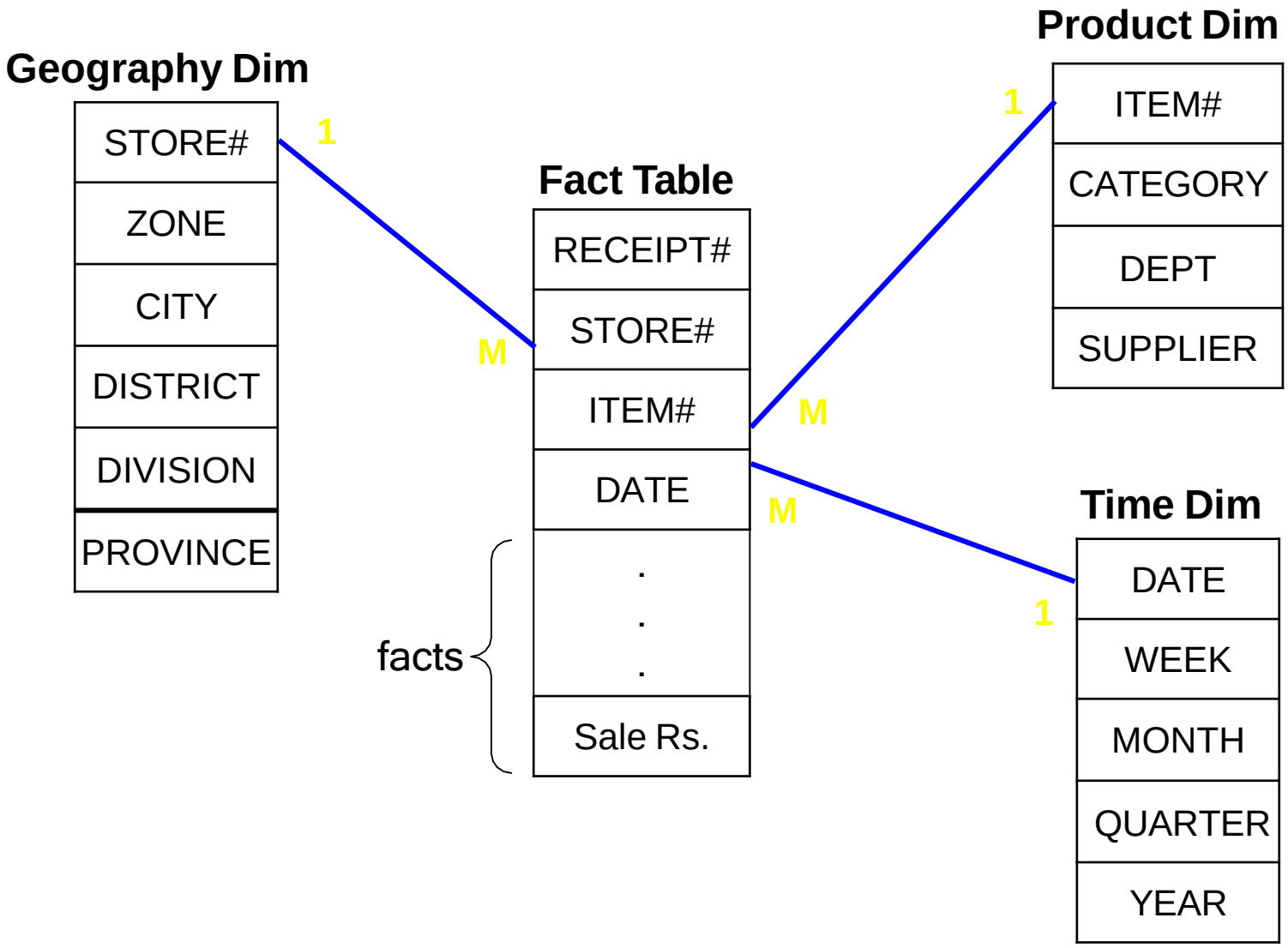


Star

“Simplified” 3NF (Retail)



Vastly Simplified Star Schema



Features of Star Schema

Dimensional hierarchies are collapsed into a single table for each dimension. Loss of Information?

A single fact table created with a single header from the detail records, resulting in:

- A vastly simplified physical data model!
- Fewer tables (thousands of tables in some ERP systems).
- Fewer joins resulting in high performance.
- Some requirement of additional space.

Process of Dimensional Modeling

The Process of Dimensional Modeling

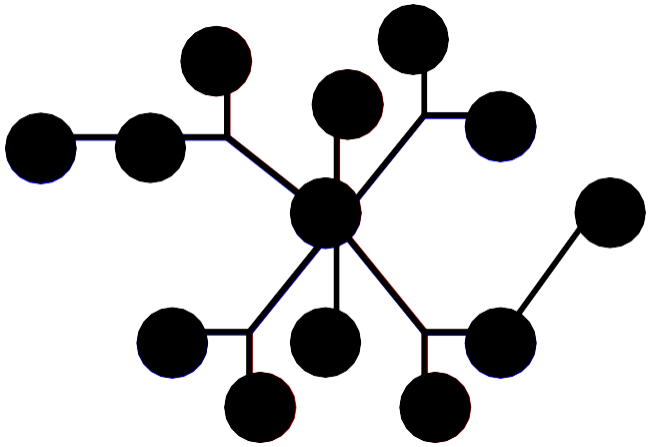
Four Step Method from ER to DM

1. Choose the Business Process
2. Choose the Grain
3. Choose the Facts
4. Choose the Dimensions

Step-1: Choose the Business Process

- A business process is a major operational process in an organization.
- Typically supported by a legacy system (database) or an OLTP.
  - Examples: Orders, Invoices, Inventory etc.
- Business Processes are often termed as Data Marts and that is why many people criticize DM as being data mart oriented.

Step-1: Separating the Process



Snow-flake

Star-1

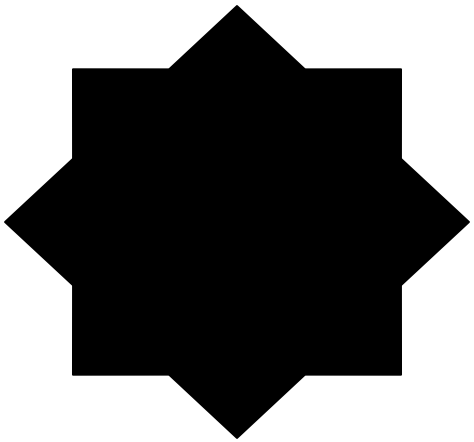
Star-2

Step-2: Choosing the Grain

- Grain is the fundamental, atomic level of data to be represented.
- Grain is also termed as the unit of analyses.
- Example grain statements
- Typical grains
  - Individual Transactions
  - Daily aggregates (snapshots)
  - Monthly aggregates
- Relationship between grain and expressiveness.
- Grain vs. hardware trade-off.

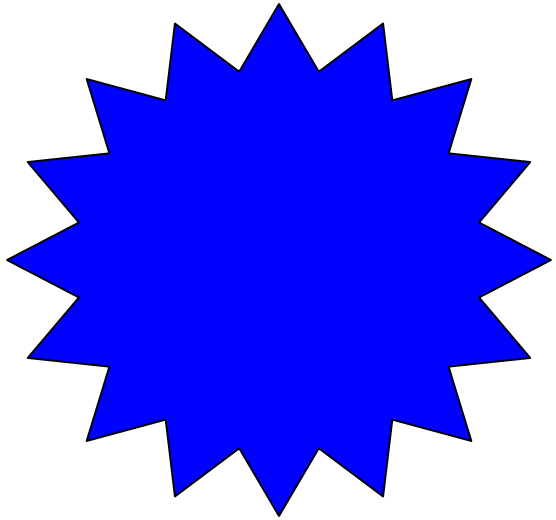
Step-2: Relationship b/w Grain

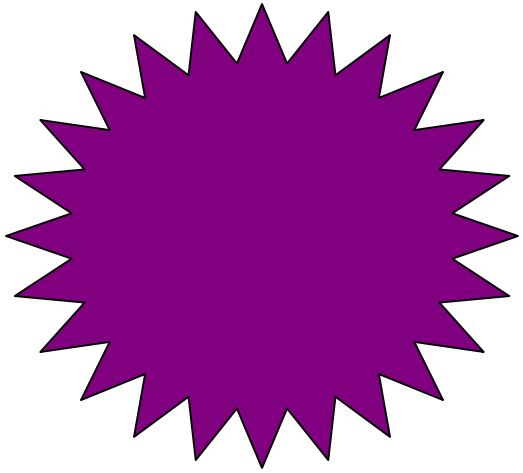
LOW Granularity → **HIGH Granularity**



Two aggregates per week
2 x 4 = 8 values

Four aggregates per week
4 x 4 = 16 values





Daily aggregates
6 x 4 = 24 values

The case FOR data aggregation

- Works well for repetitive queries.
- Follows the known thought process.
- Justifiable if used for max number of queries.
- Provides a “big picture” or macroscopic view.
- Application dependent, usually inflexible to business changes (remember lack of absoluteness of conventions).

The case AGAINST data aggregation

- Aggregation is irreversible.
  - Can create monthly sales data from weekly sales data, but the reverse is not possible.

- Aggregation limits the questions that can be answered.
  - What, when, why, where, what-else, what-next

The case AGAINST data aggregation

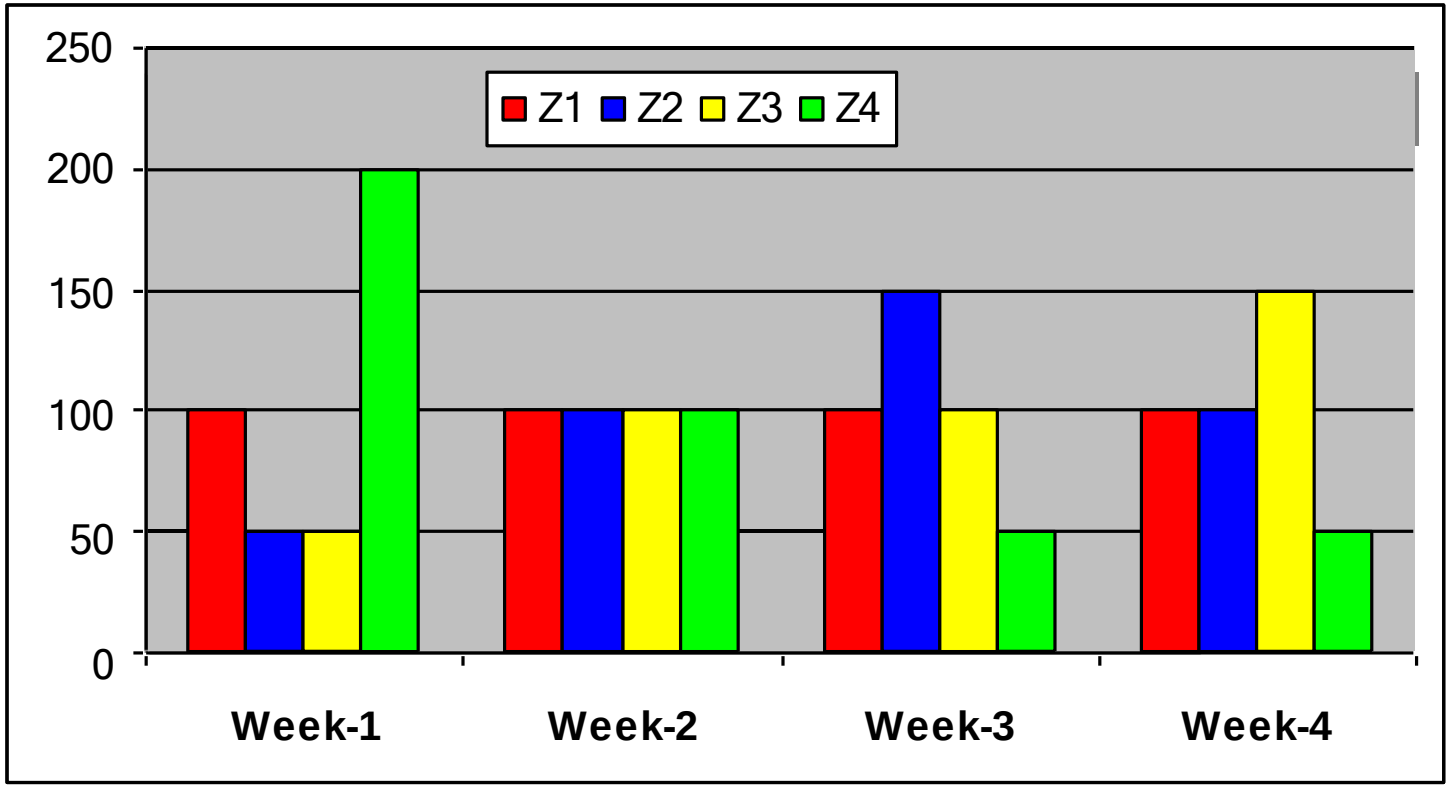
- Aggregation can hide crucial facts.

  -The average of 100 & 100 is same as 150 & 50

Aggregation hides crucial facts Example

	Week-1	Week-2	Week-3	Week-4	Average
Zone-1					100
Zone-2					100
Zone-3				te	100
Zone-4					100
Average	100	100	100	100	

Aggregation hides crucial facts chart



Z1: Sale is constant (need to work on it)

Z2: Sale went up, then fell (need of concern)

Z3: Sale is on the rise, why?

Z4: Sale dropped sharply, need to look deeply.

W2: Static sale

Step 3: Choose Facts statement

“We need monthly sales volume and cost amount by week, product and Zone”

Step 3: Choose Facts

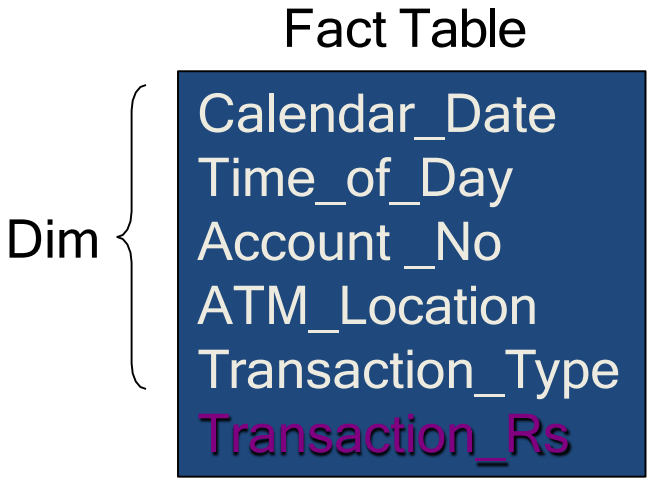
- Choose the facts that will populate each fact table record.
  - Remember that best Facts are Numeric, Continuously Valued and Additive.
  - Example: Quantity Sold, Amount etc.

Step 4: Choose Dimensions

- Choose the dimensions that apply to each fact in the fact table.
  - Typical dimensions: time, product, geography etc.
  - Identify the descriptive attributes that explain each dimension.
  - Determine hierarchies within each dimension.

Step-4: How to Identify a Dimension?

- The single valued attributes during recording of a transaction are dimensions.



Time_of_day: Morning, Mid Morning, Lunch Break etc.

Transaction_Type: Withdrawal, Deposit, Check balance etc.

Step-4: Can Dimensions be Multi-valued?

- Are dimensions ALWYS single?
  - Not really
  - What are the problems? And how to handle them

- Calendar_Date (of inspection)
- Reg_No
- Technician
- Workshop
- Maintenance_Operation

- How many maintenance operations are possible?
  - Few
  - Maybe more for old cars.

Step-4: Dimensions & Grain

- Several grains are possible as per business requirement.
  - For some aggregations certain descriptions do not remain atomic.
  - Example: Time_of_Day may change several times during daily aggregate, but not during a transaction
- Choose the dimensions that are applicable within the selected grain.

Issues of Dimensional Modeling

Additive vs. Non-Additive facts

- Additive facts are easy to work with
  - Summing the fact value gives meaningful results
  - Additive facts:
    - Quantity sold
    - Total Rs. sales
  - Non-additive facts:
    - Averages (average sales price, unit price)
    - Percentages (% discount)
    - Ratios (gross margin)
    - Count of distinct products sold

Month	Crates of Bottles Sold
May	14
Jun.	20
Jul.	24
TOTAL	58

Month	% discount
May	10
Jun.	8
Jul.	6
TOTAL	24% ← Incorrect!

Classification of Aggregation Functions

- How hard to compute aggregate from sub-aggregates?
  - Three classes of aggregates:
    - Distributive
      - Compute aggregate directly from sub-aggregates
      - Examples: MIN, MAX ,COUNT, SUM
    - Algebraic
      - Compute aggregate from constant-sized summary of subgroup
      - Examples: STDDEV, AVERAGE
      - For AVERAGE, summary data for each group is SUM, COUNT
    - Holistic
      - Require unbounded amount of information about each subgroup
      - Examples: MEDIAN, COUNT DISTINCT
      - Usually impractical for a data warehouses!

Not recording Facts

- Transactional fact tables don't have records for events that don't occur
  - Example: No records(rows) for products that were not sold.

- This has both advantage and disadvantage.
  - Advantage: Benefit of sparsity of data
    - Significantly less data to store for "rare" events
  - Disadvantage: Lack of information
  - Example: What products on promotion were not sold?

A Fact-less Fact Table

- "Fact-less" fact table
  - A fact table without numeric fact columns
  - Captures relationships between dimensions
  - Use a dummy fact column that always has value 1

Fact-less Fact Tables - Example

Examples:

- Department/Student mapping fact table
  - What is the major for each student?
  - Which students did not enroll in ANY course
- Promotion coverage fact table
  - Which products were on promotion in which stores for which days?
  - Kind of like a periodic snapshot fact

Handling Multi-valued Dimensions?

- One of the following approaches is adopted:
  - Drop the dimension.
  - Use a primary value as a single value.
  - Add multiple values in the dimension table.
  - Use “Helper” tables.

OLTP & Slowly Changing Dimensions

OLTP systems not good at tracking the past. History never changes.

OLTP systems are not “static” always evolving, data changing by overwriting.

Inability of OLTP systems to track history, purged after 90 to 180 days.

Actually don’t want to keep historical data for OLTP system.

DWH Dilemma: Slowly Changing Dimensions

The responsibility of the DWH to track the changes.

Example: Slight change in description, but the product ID (SKU) is not changed.

Dilemma: Want to track both old and new descriptions, what do they use for the key? And where do they put the two values of the changed ingredient attribute?

Explanation of Slowly Changing Dimensions

- Compared to fact tables, contents of dimension tables are relatively stable.
  - New sales transactions occur constantly.
  - New products are introduced rarely.
  - New stores are opened very rarely.
- The assumption does not hold in some cases
  - Certain dimensions evolve with time
  - e.g. description and formulation of products change with time
  - Customers get married and divorced, have children, change addresses etc.
  - Land changes ownership etc.
  - Changing names of sales regions.

Explanation of Slowly Changing Dimensions

Although these dimensions change but the change is not rapid.

Therefore called “Slowly” Changing Dimensions

Handling Slowly Changing Dimensions

- Option-1: Overwrite History
- Option-2: Preserve History
- Option-3: Create current valued field

Handling Slowly Changing Dimensions

- Option-1: Overwrite History
  - Example: Code for a city, product entered incorrectly
  - Just overwrite the record changing the values of modified attributes.
  - No keys are affected.
  - No changes needed elsewhere in the DM.
  - Cannot track history and hence not a good option in DSS.

Handling Slowly Changing Dimensions

- Option-2: Preserve History
  - Example: The packaging of a part change from glued box to stapled box, but the code assigned (SKU) is not changed.
  - Create an additional dimension record at the time of change with new attribute values.
  - Segments history accurately between old and new description
  - Requires adding two to three version numbers to the end of key. SKU#+1, SKU#+2 etc.

Handling Slowly Changing Dimensions

- Option-3: Create current valued field
  - Example: The name and organization of the sales regions change over time, and want to know how sales would have looked with old regions.
  - Add a new field called current_region rename old to previous_region.
  - Sales record keys are not changed.
  - Only TWO most recent changes can be tracked.

Pros and Cons of Handling

- Option-1: Overwrite existing value
  + Simple to implement
  - No tracking of history
- Option-2: Add a new dimension row
  + Accurate historical reporting
  + Pre-computed aggregates unaffected
  - Dimension table grows over time
- Option-3: Add a new field
  + Accurate historical reporting to last TWO changes
  + Record keys are unaffected
  - Dimension table size increases