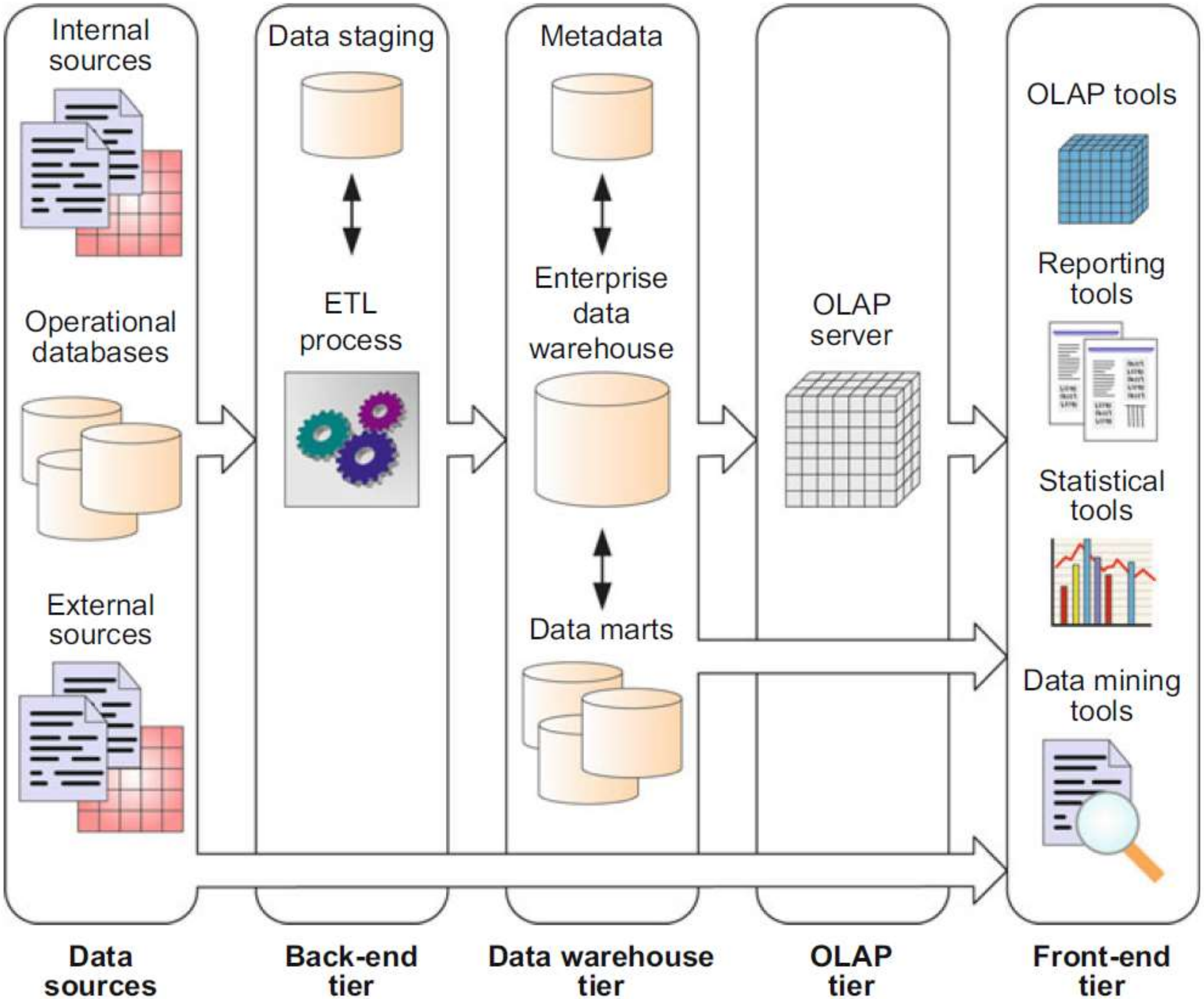
DS-306 Data Warehousing and Business Intelligence

Topic 4: Types of DW Schemas, Data Marts

Dr. Khurram Shahzad

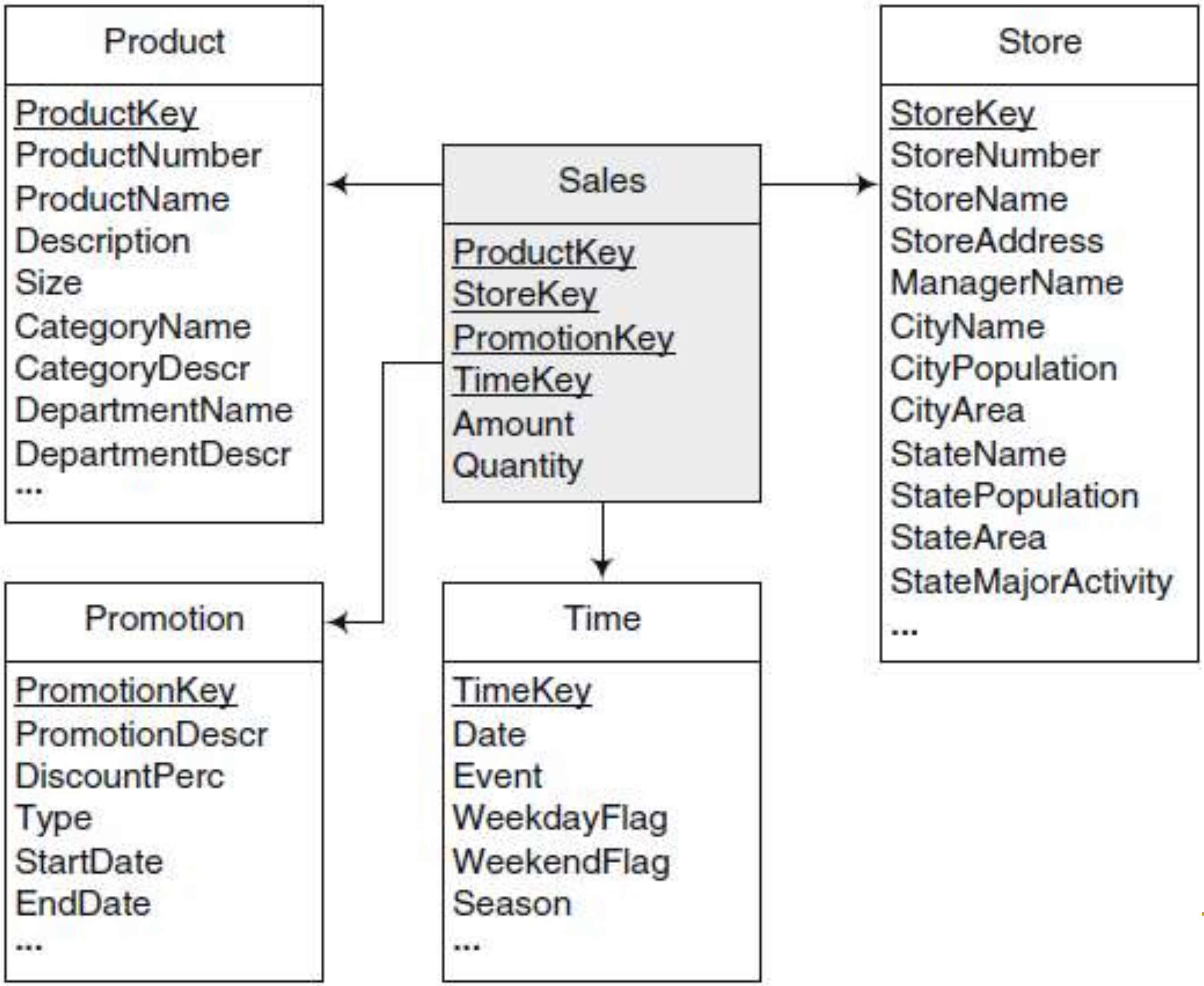
Typical DW architecture



Star Schema

- Dimension tables are **not normalized**
- Therefore, they may contain **redundant** data
- Especially, in the **presence of hierarchies**

Example Star Schema



Start Schema

- Example: Product Dimension
  - All **products** belonging to the same **category**
  - So, all the category information will be redundant

Product
ProductKey
ProductNumber
ProductName
Description
Size
CategoryName
CategoryDescr
DepartmentName
DepartmentDescr
...

Start Schema

- Example: Store Dimension
  - All stores (unique) belong to a city
    - So, all the category information will be redundant
  - All stores (unique) belongs to the same state
    - So, all the state information will be redundant

Star Schema Keys

- Primary keys (in dimension)
  - Identifying attribute in dimension table
  - Relationship attributes combine together to form P.K
- **Surrogate keys (in dimension)**
  - **Replacement of primary key**
  - **System generated**
- Foreign keys (in fact, dimension)
  - Collection of primary keys of dimension tables
- Primary key to fact table
  - Collection of P.Ks

Types of Dimensions

- Based on **size (columns and data)**
  - Large dimensions
  - Small dimensions
- Based on **changes to data**
  - Slowly Changing Dimensions
  - Rapidly Changing Dimensions

Dimensions Types: **Size**

- Based on size
  - Large dimensions
  - Small dimensions

Large Dimensions

- A dimension is considered large based on two factors
  - A large dimension is **deep** i.e., large no of rows
  - A large dimension is **wide** i.e., large no of columns
- For example, customer, product and time dimension may be gigantic
- **Multiple hierarchies**
  - Large dimensions tend to have multiple hierarchies

Example Large Dimension

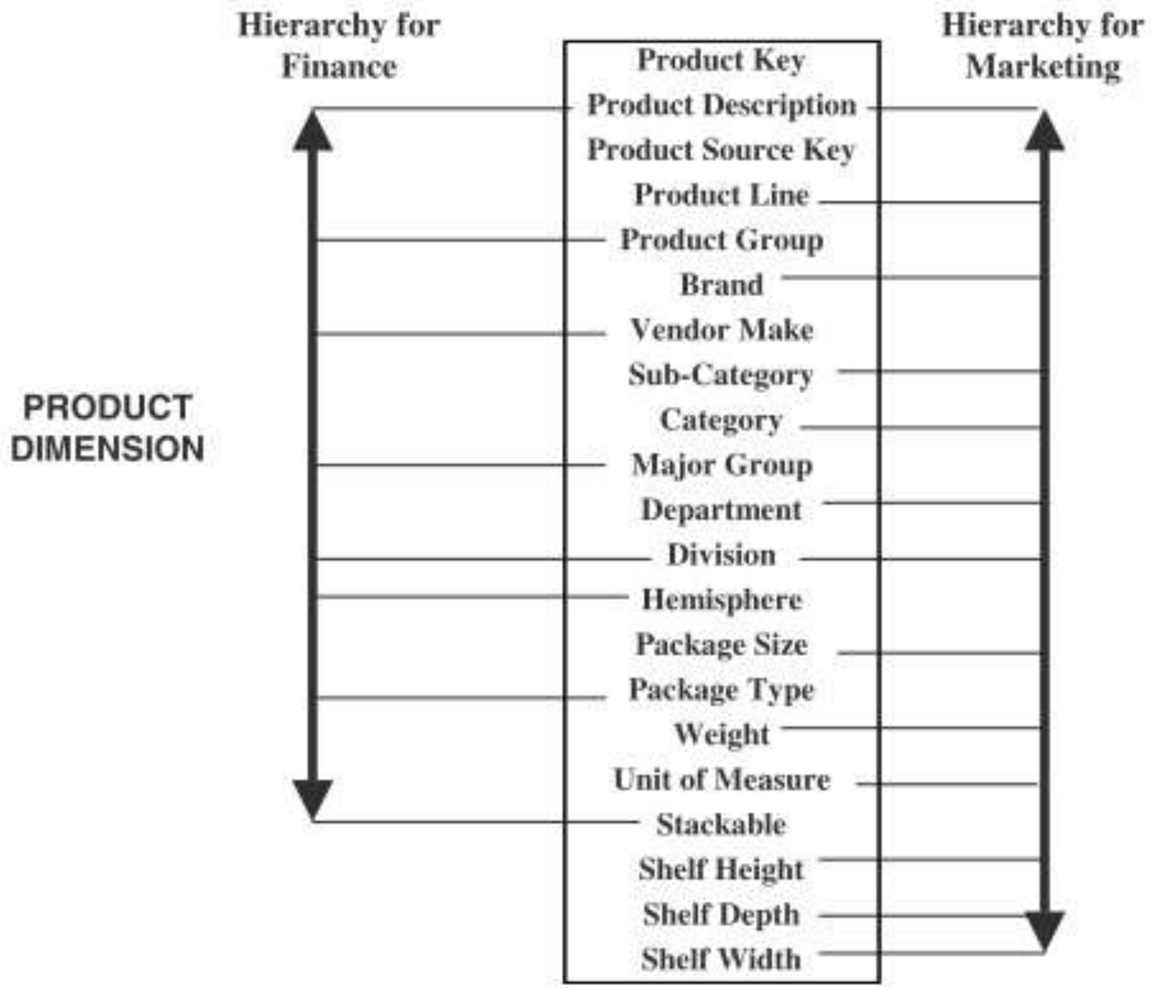


Figure 11-5 Multiple hierarchies in a large product dimension.

Small Dimensions

- Little deep
  - Fewer rows
- Little wide
  - Fewer columns
- Typically, single hierarchy
- Examples
  - Small Customer dimension
  - Small Product dimension
  - Small time dimension

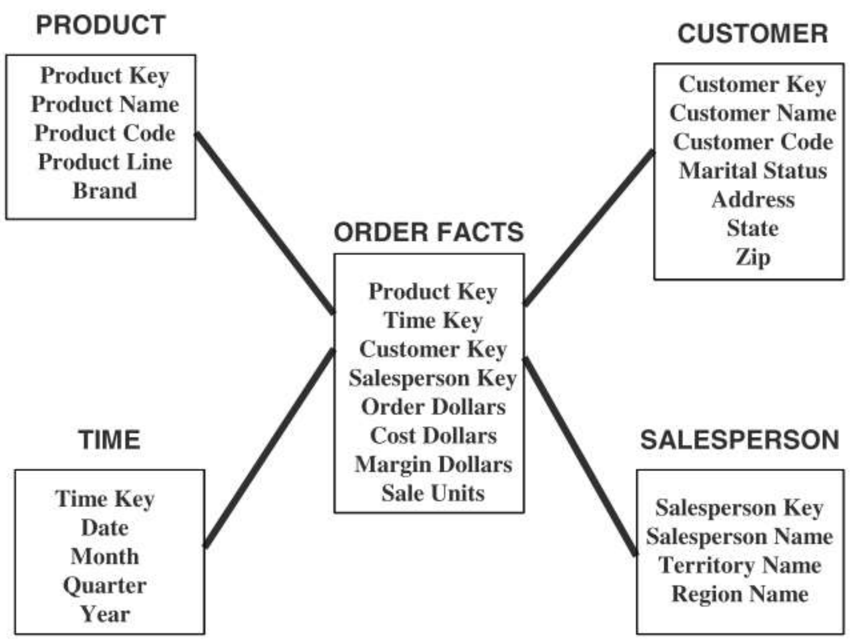
Types of Dim: **Data Changes**

- Based on changes to data of dimension
  - Slowly Changing Dimensions
  - Rapidly Changing Dimensions

- Beforehand, lets discuss **what are data changes**

Updates to dimensions

- **Fact**: sales units
- **Dimension**: Product
  - Change: A new product is added
- **Dimension**: Customer
  - New customer is added
  - Customer is relocated
  - Marital status is changed



Types of Changes

- **Type 1 Changes:** Correction of errors
- **Type 2 Changes:** History preservation
- **Type 3 Changes:** Tentative revisions

Incremental load

- DW is updated once
- Next time update / incremental update

Type 1: Correction of Error

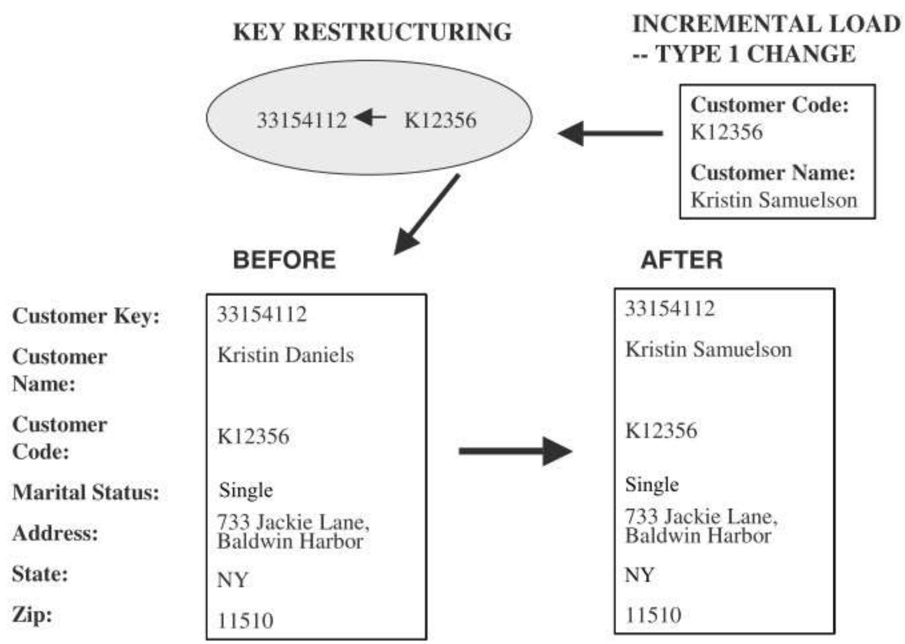
- For example, spelling of customer name is changed from Hammad Ikram to Hammad Akram
    - No need to preserve old value of Hammad "Akram"

Types 1: Correction of Error

- Kristian Daniel to Kristian Denial and the martial status is changed from single to married
  - No need to preserve old value of name
  - But change in martial status is slightly different
- Affect on result
  - Number of perfume products bought by married persons

Types 1: Correction of Errors

- Changes to DW
  - Solution, over-write old values with new values
  - Old value is not preserved
  - Key is not affected
  - Easiest to implement

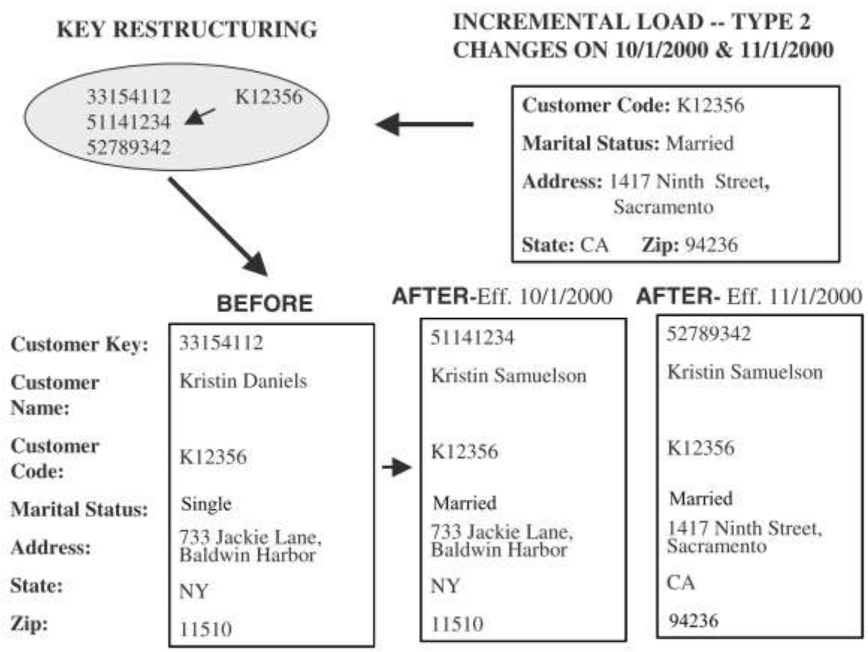


Type 2 changes: Preservation of History

- Assume that in DW an essential requirement is to track orders by marital status
  - If change happed on 1 Jan 2024 all order before that should be included in martial status single.
  - Similarly orders after that under marital status married
  - Assume city is also changed from LHR to ISL so both changes must be saved
- Principles:
  - Relate to true change in source systems
  - There is a need to preserve history in DW

Type 2 changes: Preservation of History

- Applying Type 2 changes in DW
  - There are no change to the original row in dimension table
  - Add a new dimension table row
  - The key of the original row is not affected
  - The new row is inserted with a **new surrogate key**

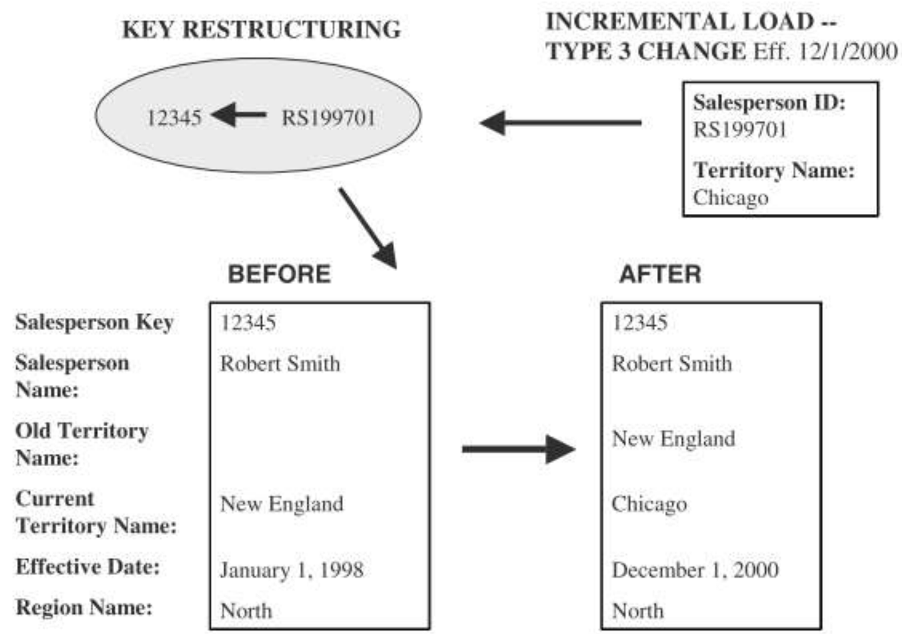


Type 3 changes: Tentative Soft Revisions

- We apply Type 2 change to a certain date (the date is a cut off point)
  - What if you need to count the orders on or after the cut-off date in both groups? This cannot be handled with type 2
  - i.e. sometimes there is a need to track both old and new values for a certain period
  - For instance, realigning territories

Type 3 changes: Tentative Soft Revisions

- Solution
- Add a new field in the dimension table
  - Push down the existing values from current to old
  - Keep the new values of the attribute in the "current" field



Summary of Types of changes

- Change1. Correction of errors
  - Problem, Name of a person is changed, to correct error
  - Solution, over-write old value
  - Old value is not saved
  - Key is not affected, Easiest to implement
- Change 2. History preservation
  - Problem, Name is changed, marital status & address
  - Solution, Inserting new row for each change
  - Change has certain affect on analysis
  - Key is not affected, new surrogate key
- Change 3. Tentative revisions
  - Problem, Old address of customer is not known
  - Solution, Insert another attribute, with date
  - New row is not needed

Recall Types of Dim: **Data Changes**

- Based on changes to data of dimension
  - Slowly Changing Dimensions
  - Rapidly Changing Dimensions

Slowly Changing Dimension

- Consideration of the changes to dimension table
  - Many dimension, though not constant over time, change slowly
  - Product key of source record does not change
  - Description and other attributes change slowly over time
  - In OLTP new values overwrite old ones
  - Overwriting not always appropriate option in DW

Rapidly Changing Dimension

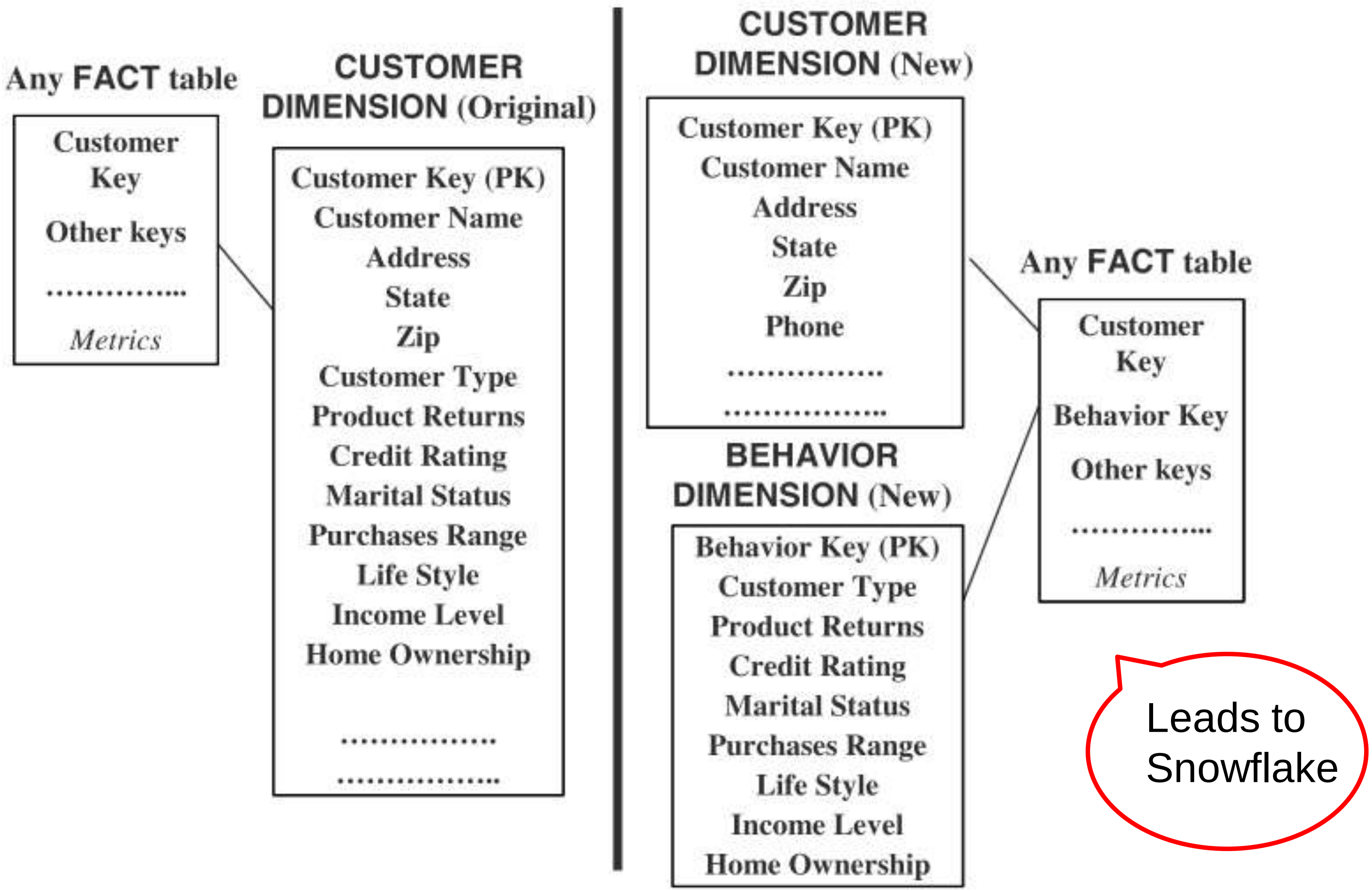
- For Type 2 changes, a new row is created with new attribute value
  - Preserve the history
- What if the change occurs too many times?
- This dimension is no longer a slowly changing dimension
- Particularly, in Large dim. frequent changes cannot be handled with any type of change

Rapidly changing dimensions

- But consider a large customer dimension, where millions of customers may exist
  - But significant attributes in a customer dimension may change many times in a year (rapidly changing dimension)
- In the case if dimension table is too large and changing too rapidly

Solution????

Rapidly changing dimensions



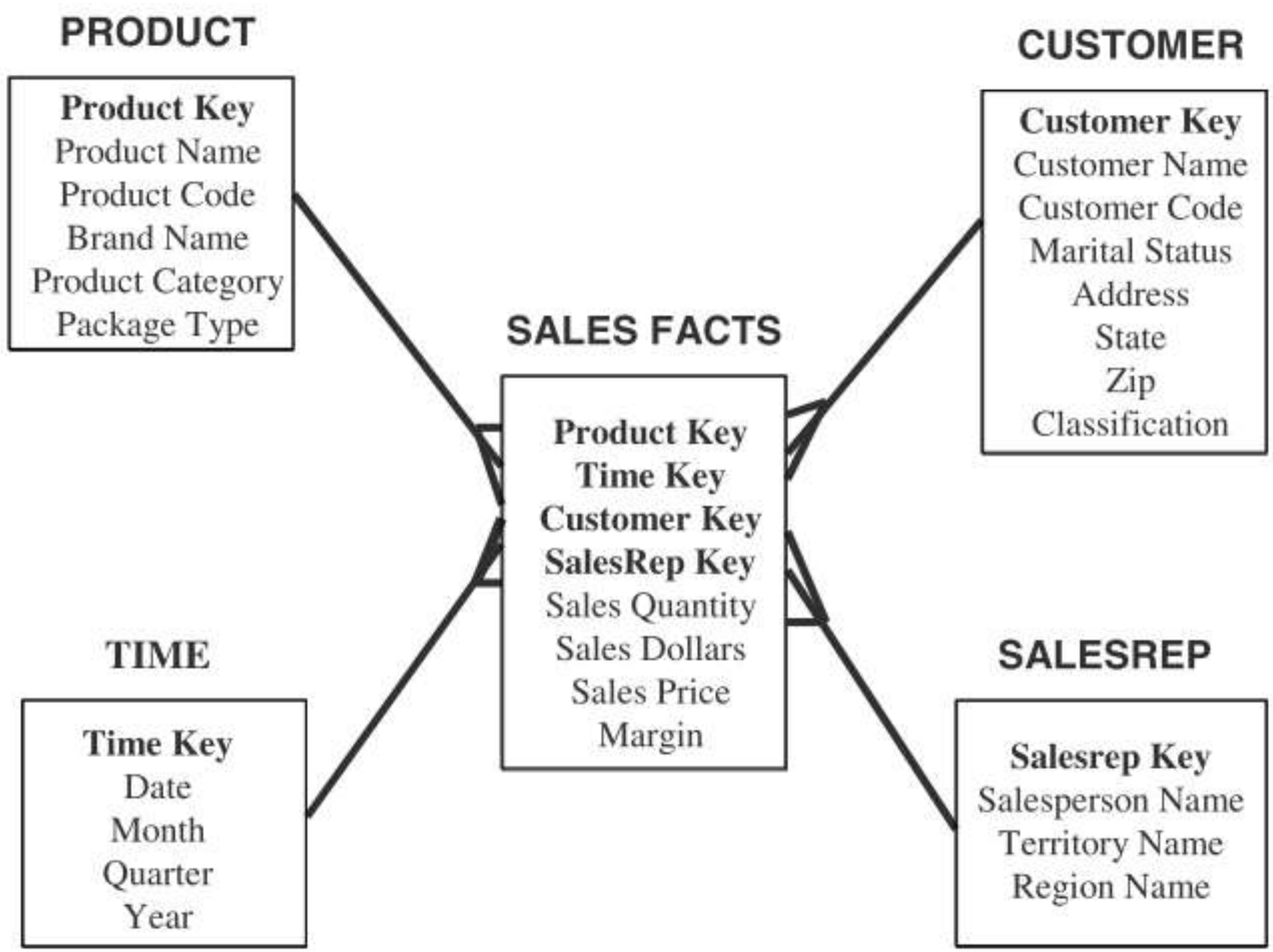
Types of Dimensional Models

- Star Schema (discussed)
- Snowflake Schema
- Starflake Schema
- Constellation Schema

Snowflake Schema

- Snowflake avoids the redundancy of star schemas by **normalizing** the dimension tables
- Therefore, a dimension is represented by several tables **related by referential integrity**
- Referential integrity constraints is also **between fact table and the dimension** tables at the fines level of details

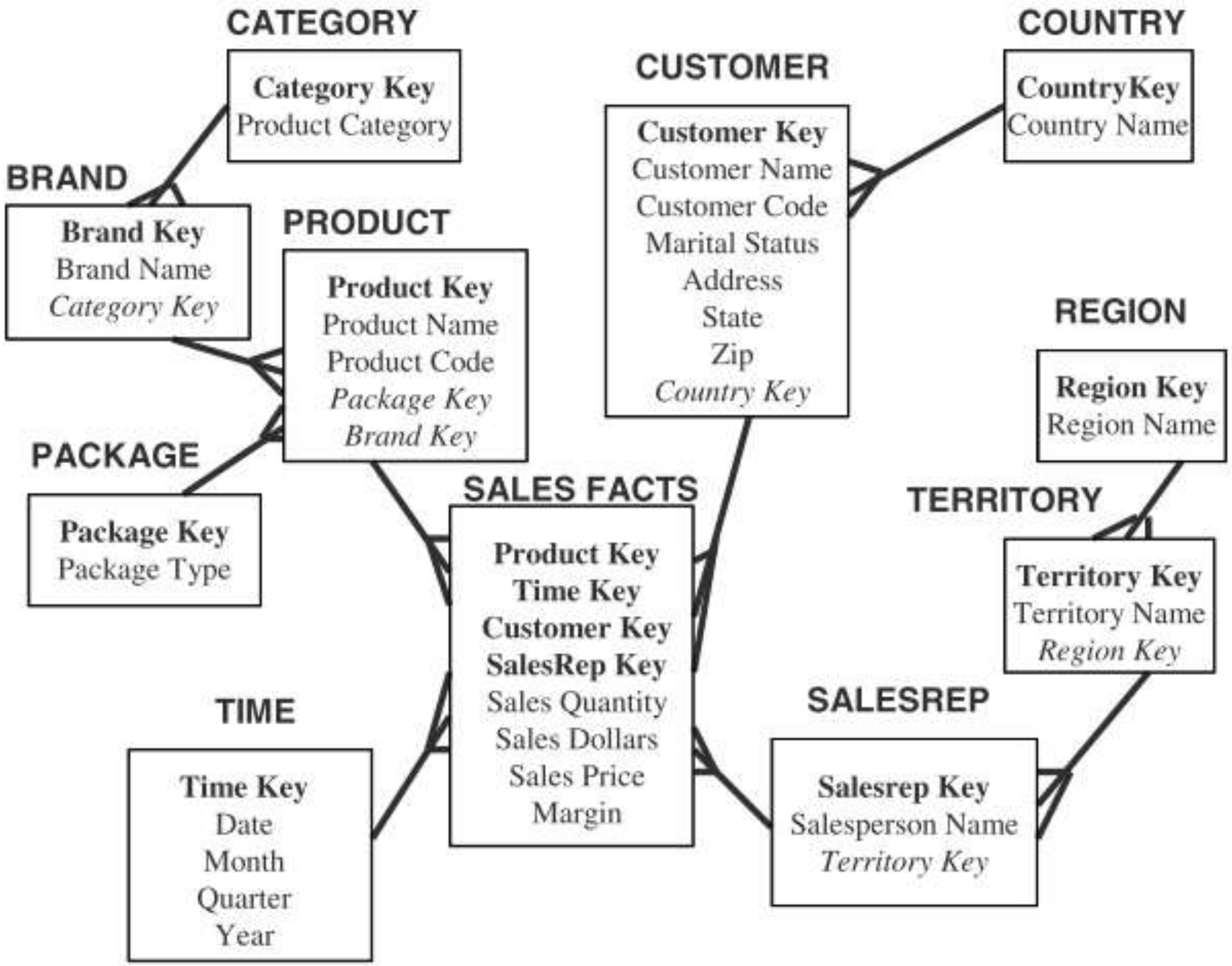
From Star to Snowflake Schema



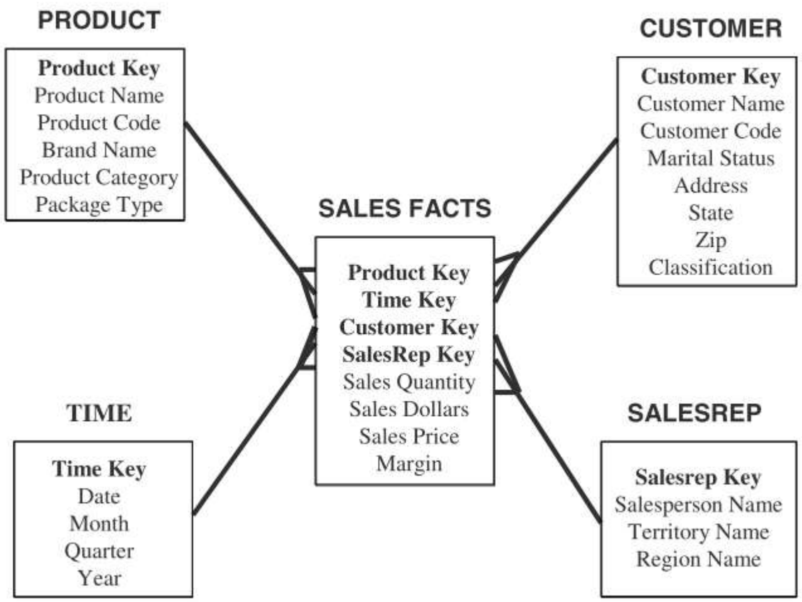
Normalization of Dimensions

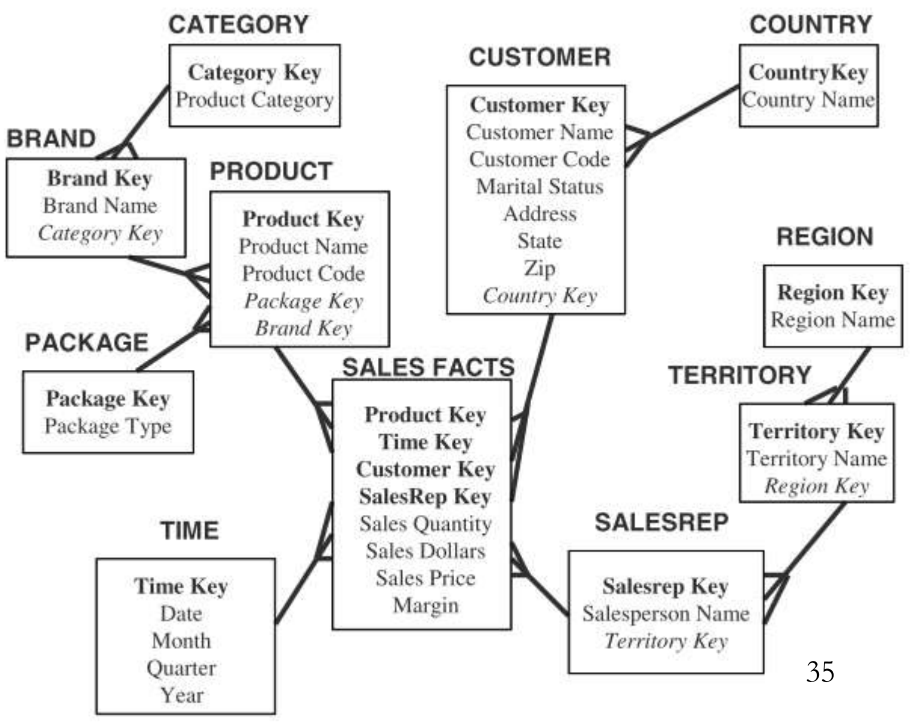
- 1st Normal Form
- 2nd Normal Form
- 3rd Normal Form

From Star to Snowflake Schema



From Star to Snowflake Schema

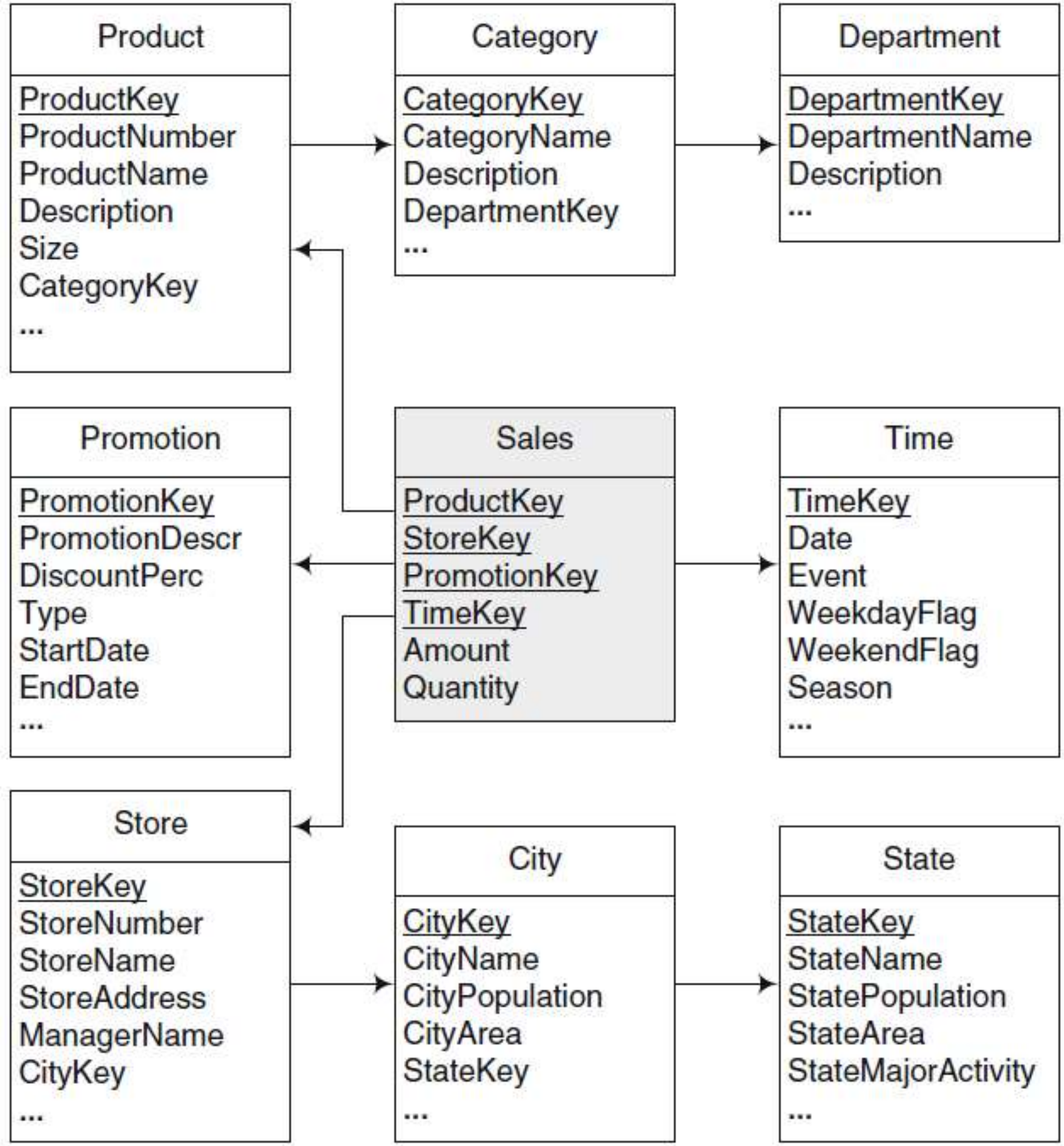




Search Reduction in Snowflake

- Assume there are 500,000 product dimension rows
- And 500 product brands, 10 product categories
- Query about 1 product category
  - Searching will be performed on 500,000 rows
  - But, if it partially normalized by product brands and product category
  - Initial search, 10 rows

Another Example Snowflake



Snowflake: Advantages & Disadvantages

- Advantages
  - Small saving in storage
  - Normalized structures are easier to update
- Disadvantages
  - Schema is less intuitive (complex)
  - Ability to browse through content difficult
  - Degraded query performance due to joins

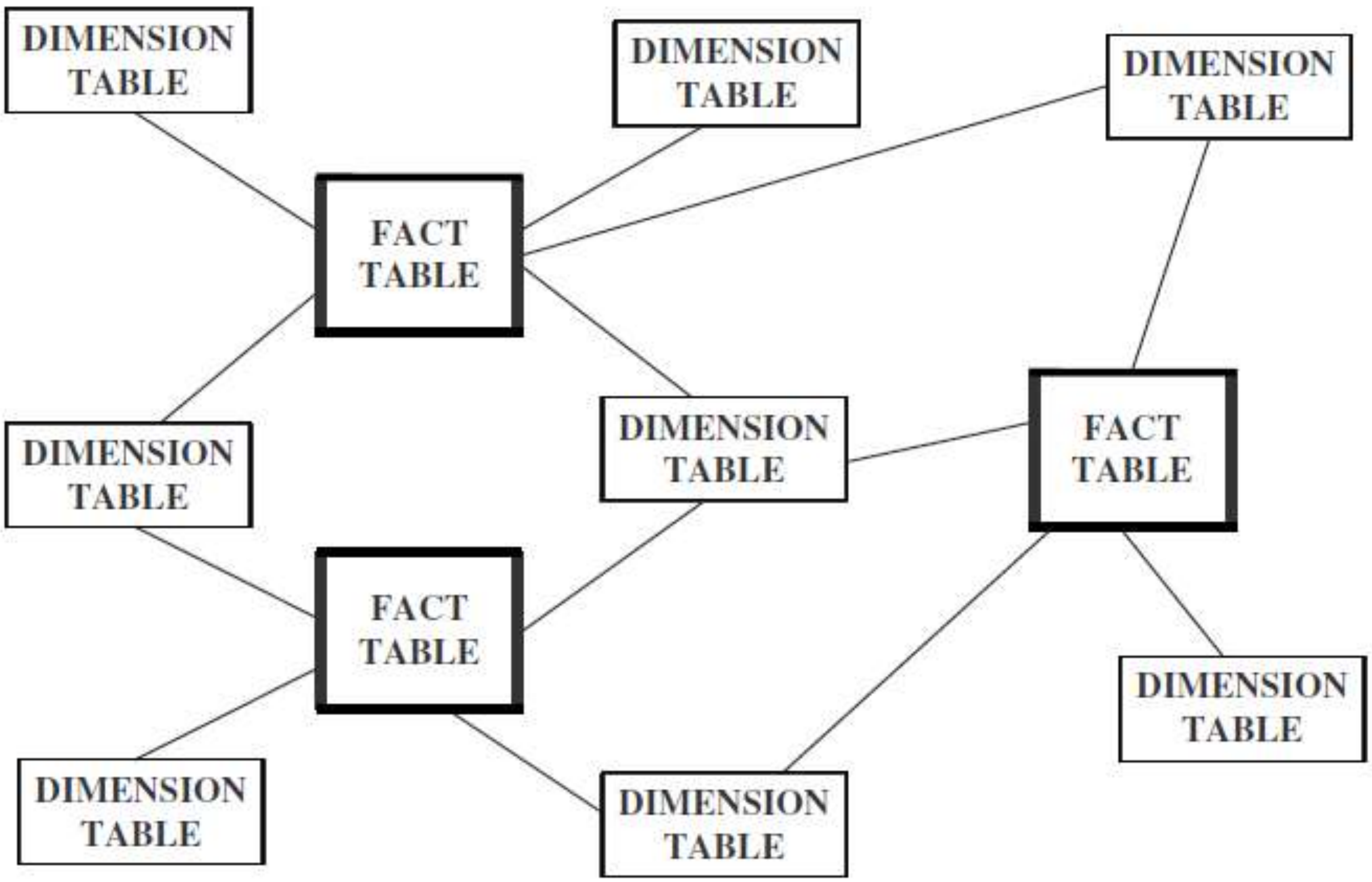
Starflake Schema

- A combination of star and snowflake.
- Some dimensions are normalized while others are not

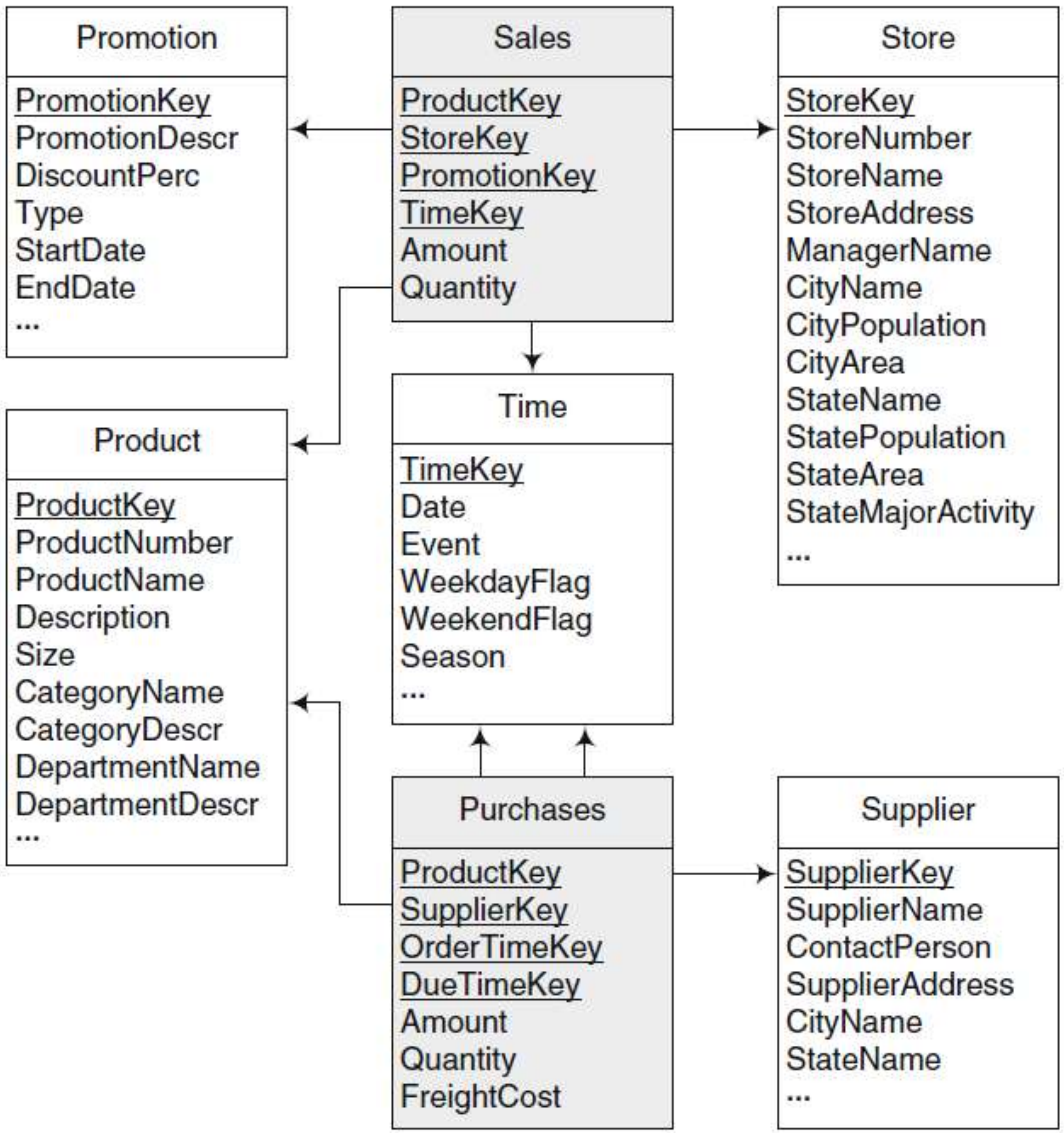
Constellation Schema

- Schema has multiple fact tables that share dimension tables

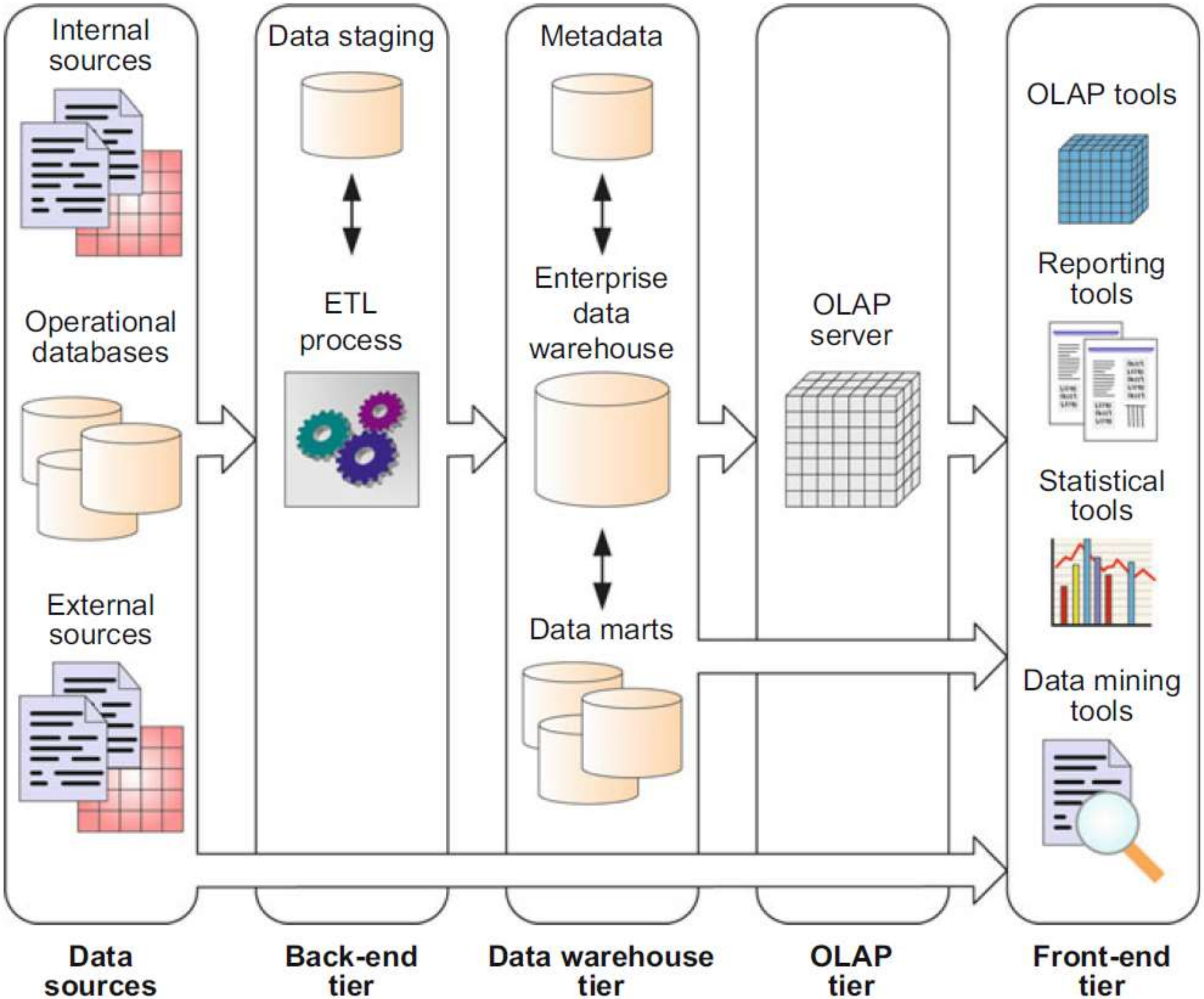
Constellation Schema



Constellation Schema Example



Typical DW architecture



Data Marts

- DW aims at analyzing the data of **entire** organization
- Sometimes, departments or division requires **portion** of the data warehouse
- They have **specialized needs**
  - Sales department needs sales data
  - HR department need demographic data and employee data
- Departmental data warehouses are called data marts

Data Marts

- Marts can be **derived** from DW or from data sources
- Can have Star or Snowflake schemas
- DW can be seen as a collection of data marts
- Marts are easier to build than an enterprise DW

Data Marts

- Two types of design approach
  - Bottom-up approach
    - Marts $\longrightarrow$ DW
  - Top-down approach
    - DW $\longrightarrow$ Marts

Top-down Approach

- Advantages
  - A truly corporate effort, enterprise view of data
  - Inherently architecture – not a union of marts
  - Single, central storage
- Disadvantages
  - Takes longer to build
  - High risk of failure
  - Needs high level of cross-functional skills

Bottom-down Approach

- Advantages
  - Faster and easier implementation
  - Favorable return on investment for PoC
  - Inherently incremental
  - Allows project team to learn and grow
- Disadvantages
  - Data mart has its own narrow view of data
  - Redundant data in every mart
  - Perpetuates inconsistent and irreconciled data

When to use DW or Mart?

- Questions that determine DW or Data Marts
  - Top-down or bottom-up approach
  - Enterprise-wide or departmental
  - Which first – data warehouse or data mart
  - Build pilot or go with full-fledge implementation
  - Dependent or independent data marts

DW vs Marts Summary

DATA WAREHOUSE	DATA MART
◆ Corporate/Enterprise-wide ◆ Union of all data marts ◆ Data received from staging area ◆ Queries on presentation resource ◆ Structure for corporate view of data ◆ Organized on E-R model	◆ Departmental ◆ A single business process ◆ Star-join (facts & dimensions) ◆ Technology optimal for data access and analysis ◆ Structure to suit the departmental view of data

Case Study: Draw Snowflake