DIMENSIONAL MODELING

Why we need to go for Dimensional Model?

- To overcome ***performance issues*** for large queries in the data warehouse, we use dimensional models.

- The dimensional modeling approach provides a way to improve query performance for summary reports without affecting data integrity.

Dimensional model

- **The dimensional model consists of two types of tables having different characteristics.**

- **They are:**

  - **Dimension table**
  - **Fact table**

What is a dimensional model?

- ***A dimensional model is also commonly called a "star schema" .***

- **This type of model is very popular in data warehousing because it can provide much better query performance, especially on very large queries, than an E/R model.**

- **However, it also has the major benefit of being easier to understand. It consists, typically, of a large table of facts (known as a fact table), with a number of other tables surrounding it that contain descriptive data, called dimensions**

Dimensions

- A dimension is a collection of members or units of the same type of views.
- Dimensions determine the contextual background for the facts.
- Dimensions represent the way business people talk about the data resulting from a business process, e.g., who, what, when, where, why, how ???

What is a Fact ?

- A fact is a collection of related data items, consisting of measures and context data.
- Each fact typically represents a business item, a business transaction, or an event that can be used in analyzing the business or business process.
- Facts are measured, "continuously valued", rapidly changing information. Can be calculated and/or derived.
- **Granularity**

The level of detail of data contained in the data warehouse

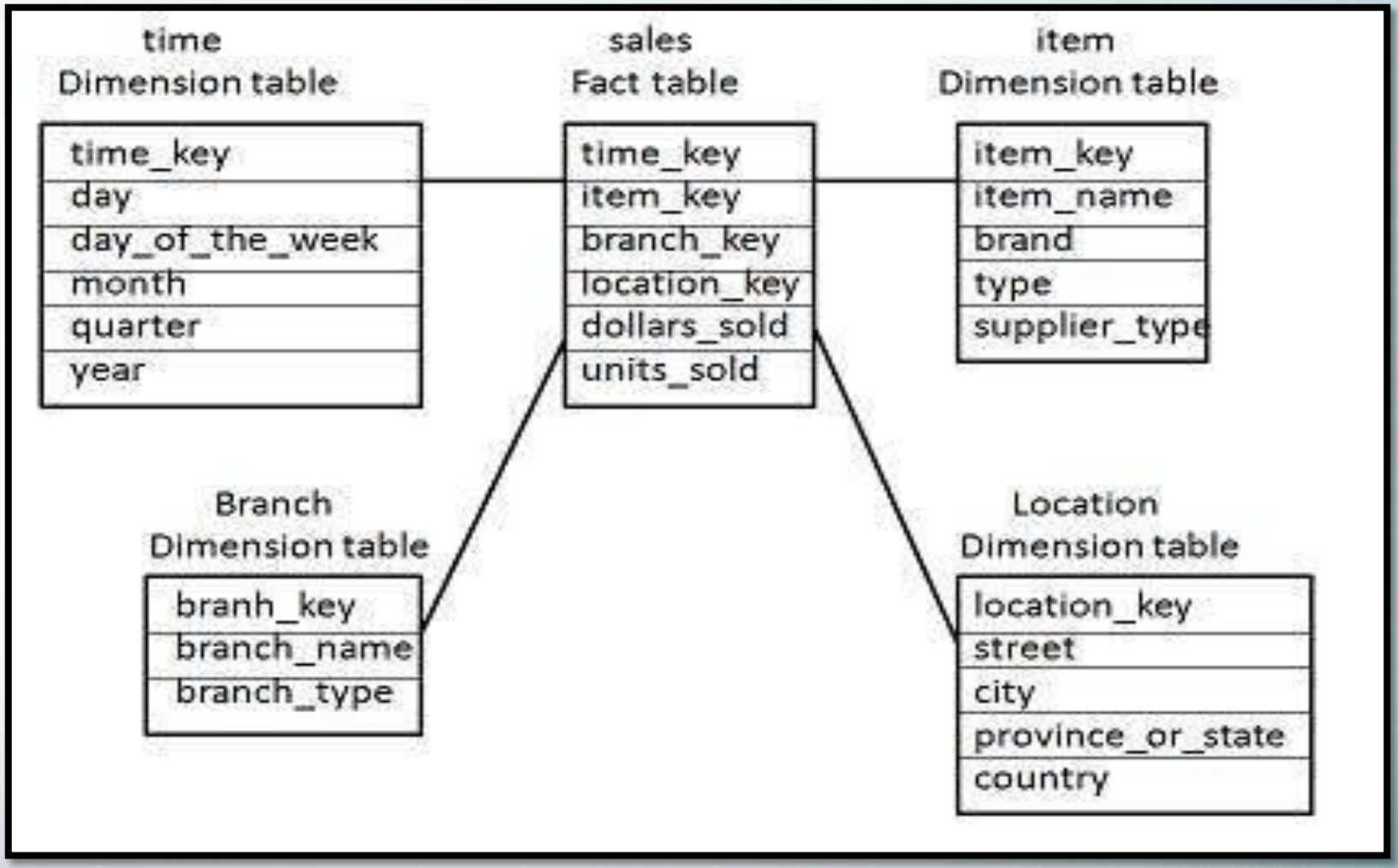
e.g. Daily item totals by product, by store

TYPES OF SCHEMAS

Star Schema Design

- **Single fact table surrounded by denormalized dimension tables**
- **The fact table primary key are the foreign keys (i.e. primary keys of dimension tables)**
- **Fact table contains transaction type information.**
- **Easily understood by end users, more disk storage required**

EXAMPLE OF STAR SCHEMA

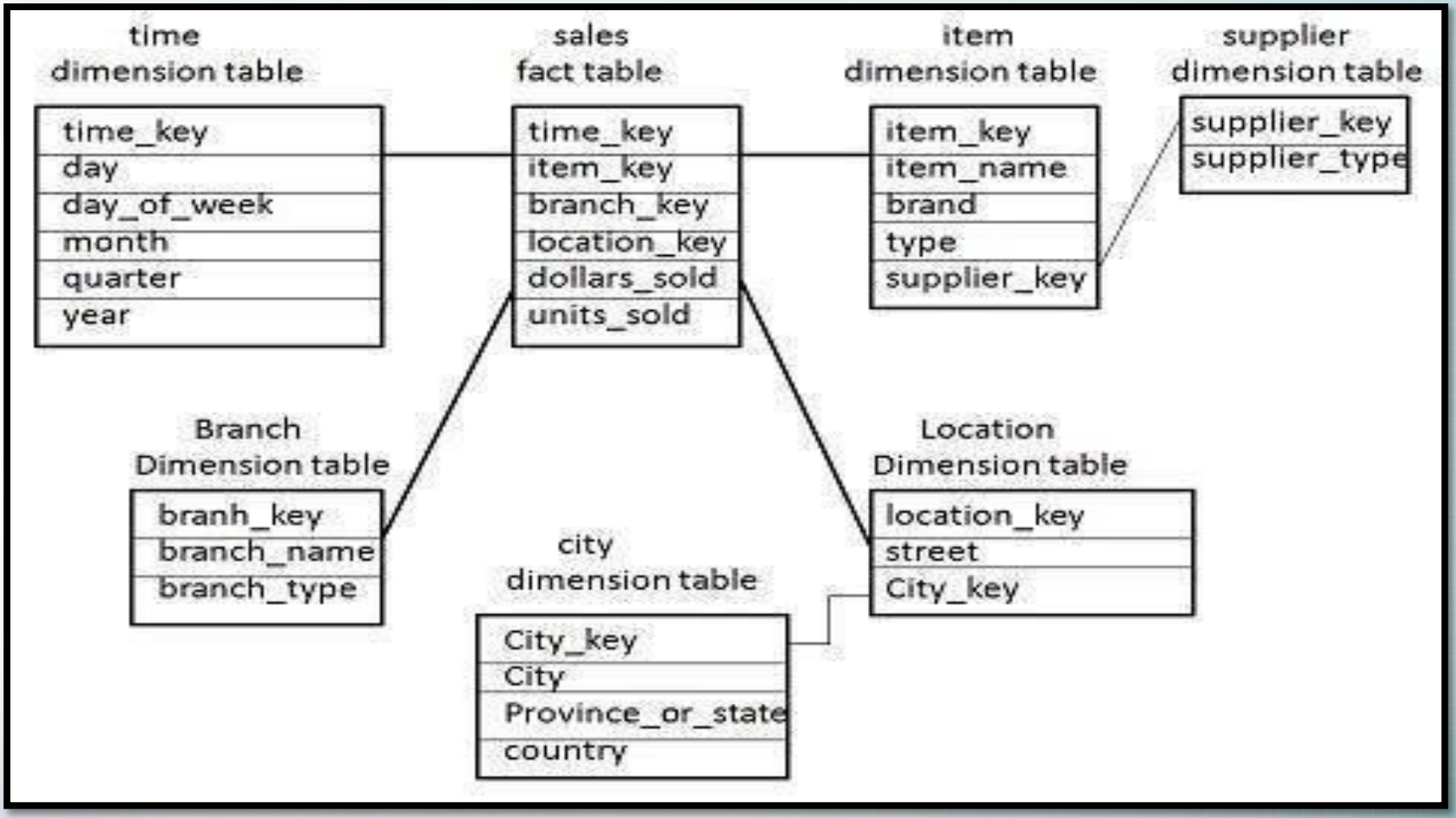


Snowflake Schema

- Single fact table surrounded by normalized dimension tables
- Normalizes dimension table to save data storage space.
- Here in Snow Flake Schema dimensions become very very large
- Less intuitive, slower performance due to joins

May want to use both approaches, especially if supporting multiple end-user tools.

Snowflake Schema

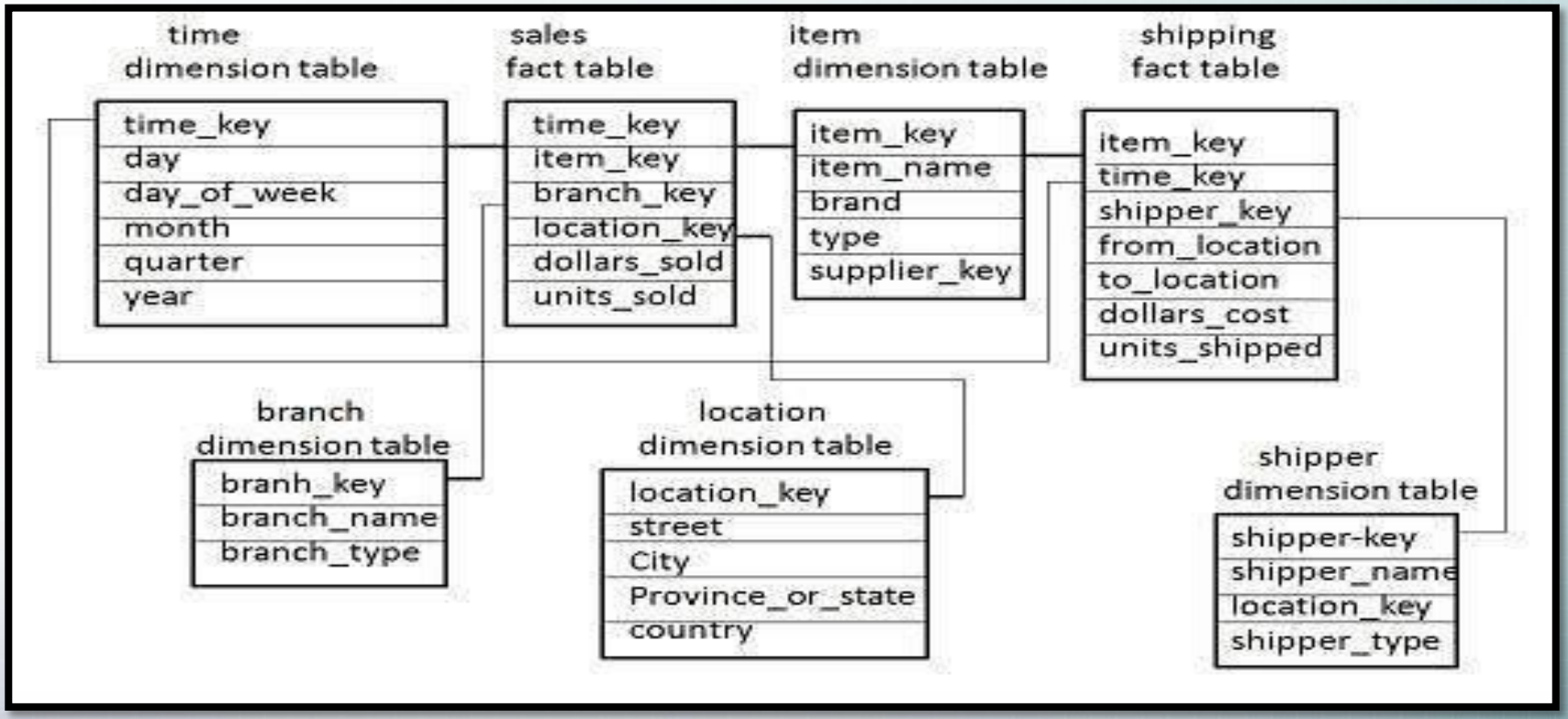


Snowflake Schema Disadvantages

- **Normalization of dimension makes it difficult for user to understand**
- **Decreases the query performance because it involves more joins**

Fact Constellation

- Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation



TYPES OF MEASURES

Types of Measures / Facts

- **There are three types of facts:**
  - **Additive**: Additive facts are facts that can be summed up through all of the dimensions in the fact table.
  - **Semi-Additive**: Semi-additive facts are facts that can be summed up for some of the dimensions in the fact table, but not the others.
  - **Non-Additive**: Non-additive facts are facts that cannot be summed up for any of the dimensions present in the fact table.

Let us use examples to illustrate each of the three types of facts

- **The first example assumes that we are a retailer, and we have a fact table with the following columns:**

Date
Store
Product
Sales_Amount

ADDITIVE FACTS

- **The purpose of this table is to record the sales amount for each product in each store on a daily basis. Sales_Amount is the fact.**

- **In this case, Sales_Amount is an additive fact, because you can sum up this fact along any of the three dimensions present in the fact table -- date, store, and product.**

- **For example, the sum of Sales_Amount for all 7 days in a week represents the total sales amount for that week.**

SEMI – ADDITIVE FACTS

- **Say we are a bank with the following fact table:**

Date
Account
Current_Balance
Profit_Margin

The purpose of this table is to record the current balance for each account at the end of each day, as well as the profit margin for each account for each day. Current_Balance and Profit_Margin are the facts. Current_Balance is a semi-additive fact, as it makes sense to add them up for all accounts (what's the total current balance for all accounts in the bank?)

NON – ADDITIVE FACTS

- But it does not make sense to add them up through time (adding up all current balances for a given account for each day of the month does not give us any useful information).

- Profit_Margin is a non-additive fact, for it does not make sense to add them up for the account level or the day level.

Types of Fact Tables

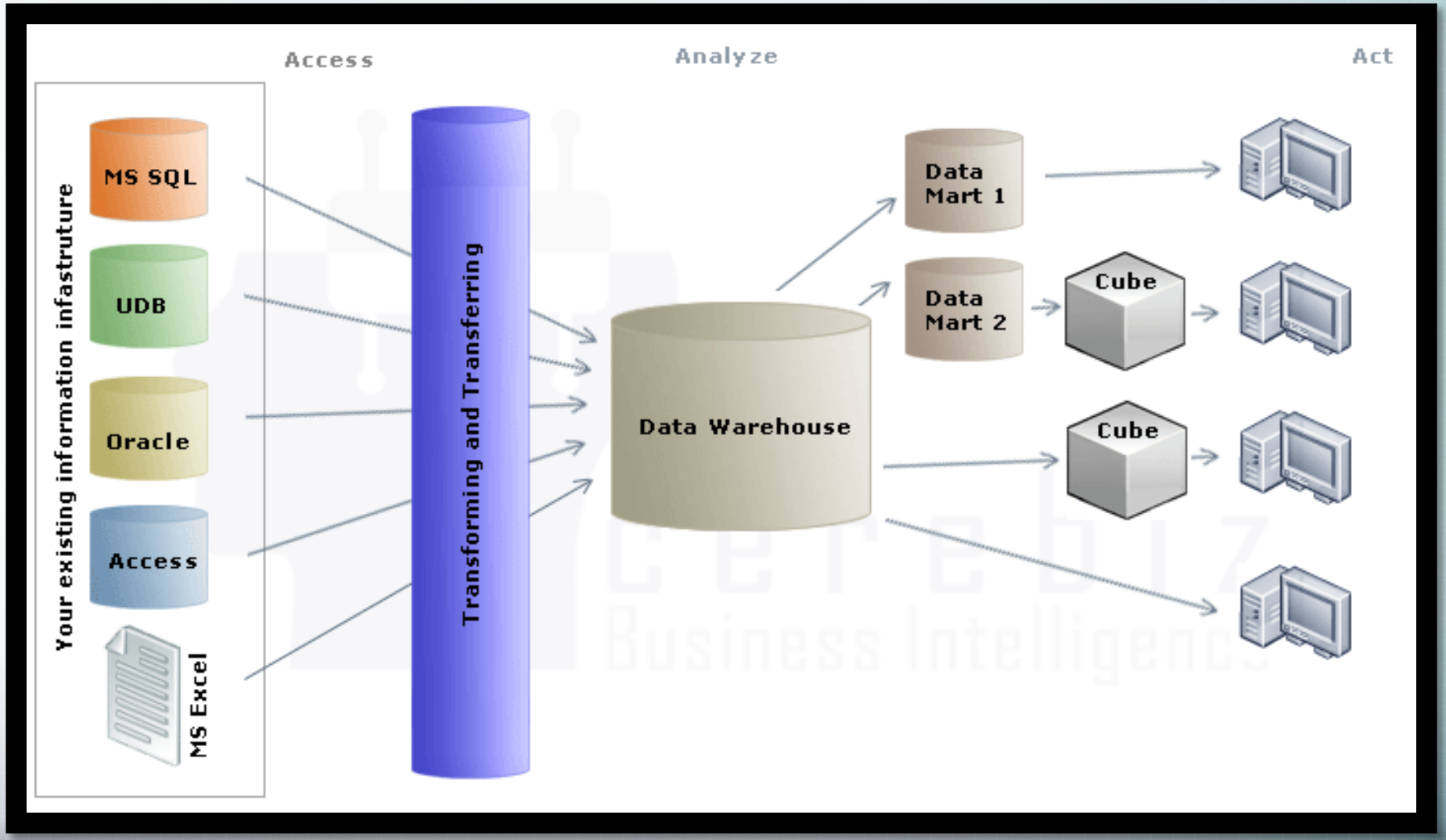
- Based on the above classifications, there are two types of fact tables:
  - Cumulative: This type of fact table describes what has happened over a period of time. For example, this fact table may describe the total sales by product by store by day.
  - The facts for this type of fact tables are mostly additive facts. The first example presented here is a cumulative fact table.

Types of Fact Tables

- **Snapshot**: This type of fact table describes the state of things in a particular instance of time, and usually includes more semi-additive and non-additive facts.

- The second example presented here is a snapshot fact table.

DWH ARCHITECTURE



Properties of DWH

- **The Properties of Data warehouse are as follows:**
  - **Subject- Oriented**
  - **Integrated**
  - **Non-Volatile**
  - **Time Variant**

DIFFERENCE B/W ER MODEL AND DIMENSIONAL MODEL

ER - MODELLING	DM - MODELLING
A view of data from data processing	A view of data from business processing
Contains both Logical and Physical model	Contains only Physical Model
It process normalized data	It process denormalized data
It is used for OLTP systems that uses 1st or 2nd or 3rd Normal Form	It is used for DWH Systems that uses 3rd Normal Form
It is not mapped for creating Schemas	It is mapped for creating Schemas

DIFFERENCE B/W ER MODEL AND DIMENSIONAL MODEL

ER - MODELLING	DM - MODELLING
DATA: Uses Current Data	**DATA**: Uses Historical Data
USER: More than 1000 users	**USER**: Only Top Management
SIZE: MB to GB	**SIZE**: GB to TB
PROCESS: Normalization	**PROCESS**: Denormalization
DATA STORAGE: Volatile	**DATA STORAGE**: Non – Volatile
Removes data Redundancy, Ensures Data Consistency and Expresses relationship b/w entities.	Captures Critical Measures, Views along Dimensions and Useful to business users.

CUBE’s in DWH

CUBE's IN DWH

- An OLAP **cube** is a multidimensional database that is optimized for **data warehouse** and online analytical processing (OLAP) applications.
- An OLAP **cube** is a method of storing **data** in a multidimensional form, generally for reporting purposes.
- In OLAP **cubes**, **data** (measures) are categorized by dimensions.

CUBE'S

- **Cubes in a data warehouse are stored in three different modes.**

- ***A relational storage model is called Relational Online Analytical Processing mode or ROLAP, while a Multidimensional Online Analytical processing mode is called MOLAP.***

- ***When dimensions are stored in a combination of the two modes then it is known as Hybrid Online Analytical Processing mode or HOLAP.***

Implementation Types of OLAP Servers

- **We have four types of OLAP servers:**
  - **Relational OLAP (ROLAP)**
  - **Multidimensional OLAP (MOLAP)**
  - **Hybrid OLAP (HOLAP)**

RELATIONAL OLAP

- The underlying data in this model is stored in relational databases. Since the data is stored in relational databases this model gives the appearance of traditional OLAP's slicing and dicing functionality.
- Advantages of this model is it can handle a large amount of data and can leverage all the functionalities of the relational database.
- The disadvantages are that the performance is slow.

Multidimensional OLAP

- **Multidimensional OLAP:**

  - **MOLAP uses array-based multidimensional storage engines for multidimensional views of data.**

  - **With multidimensional data stores, the storage utilization may be low if the data set is sparse.**

  - **Therefore, many MOLAP server use two levels of data storage representation to handle dense and sparse data sets.**

Hybrid OLAP

- **Hybrid OLAP (HOLAP)**

  - **Hybrid OLAP is a combination of both ROLAP and MOLAP.**

  - **It offers higher scalability of ROLAP and faster computation of MOLAP.**

  - **HOLAP servers allows to store the large data volumes of detailed information. The aggregations are stored separately in MOLAP store.**

NEED FOR MULTIDIMENSIONAL OLAP

Multidimensional OLAP

- This is the traditional mode in OLAP analysis. In MOLAP data is stored in form of multidimensional cubes and not in relational databases.
- The advantages of this mode is that it provides excellent query performance and the cubes are built for fast data retrieval.
- All calculations are pre-generated when the cube is created and can be easily applied while querying data. The disadvantages of this model are that it can handle only a limited amount of data.
- Since all calculations have been pre-built when the cube was created, the cube cannot be derived from a large volume of data

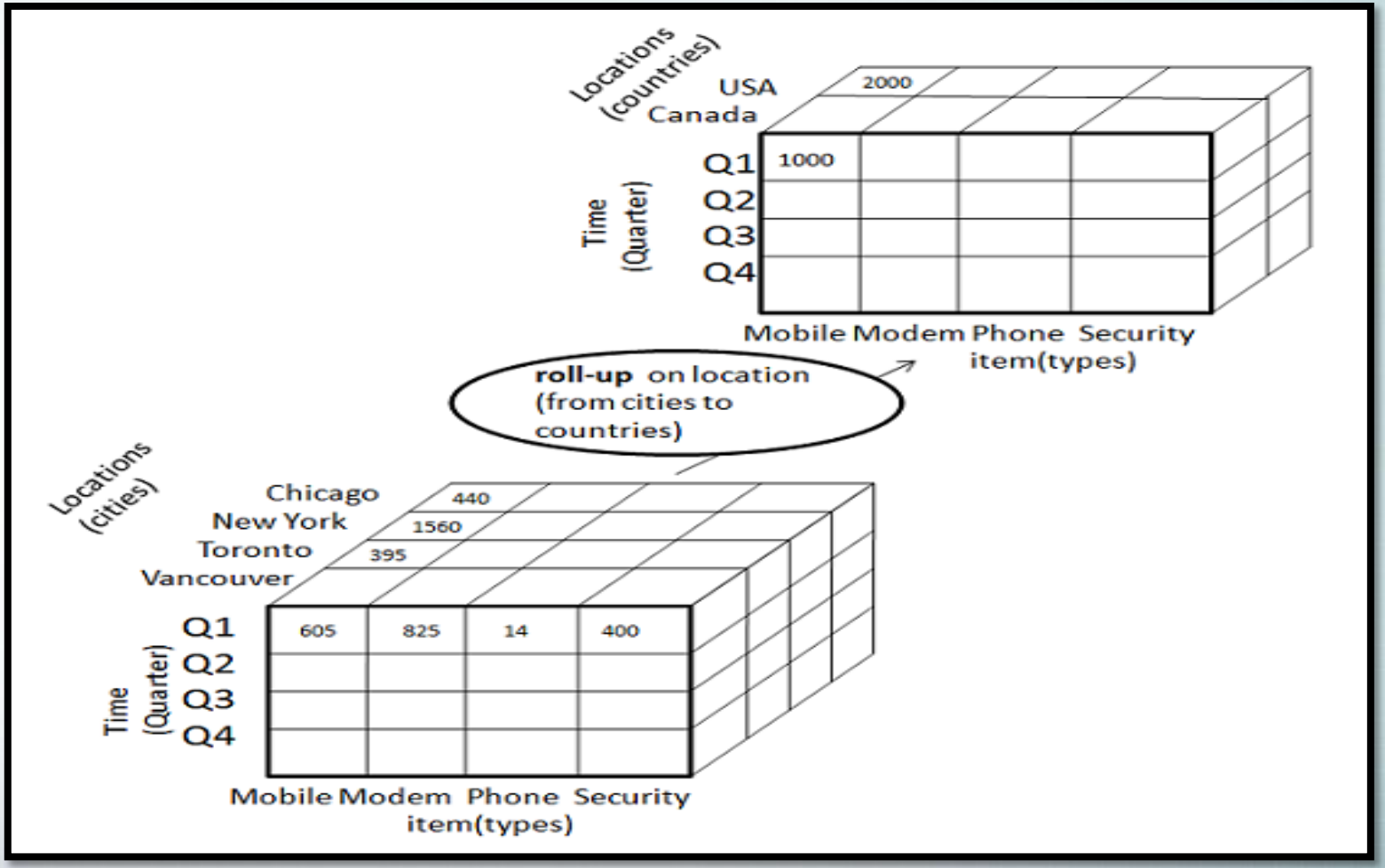
OLAP Operations

- **Here is the list of OLAP operations:**
  - **Roll-up**
  - **Drill-down**
  - **Slice and dice**
  - **Pivot (rotate)**

Roll-up

- **Roll-up performs aggregation on a data cube in any of the following ways:**
  - **By climbing up a concept hierarchy for a dimension**
  - **By dimension reduction**

Roll – Up Example:



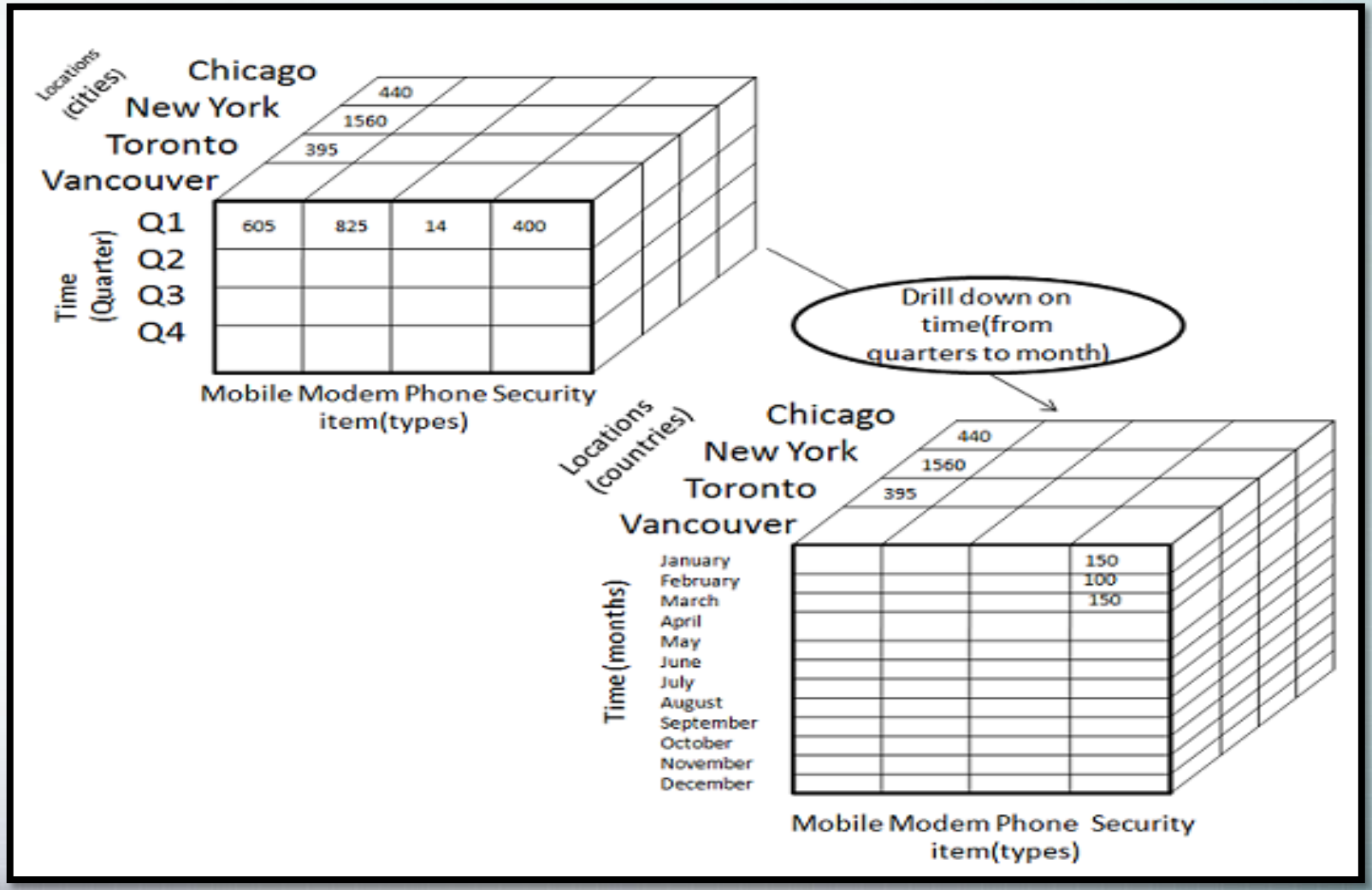
ROLL – UP

- Roll-up is performed by climbing up a concept hierarchy for the dimension location.
- Initially the concept hierarchy was "street < city < province < country".
- On rolling up, the data is aggregated by ascending the location hierarchy from the level of city to the level of country.
- The data is grouped into cities rather than countries.
- When roll-up is performed, one or more dimensions from the data cube are removed.

Drill-down

- **Drill-down is the reverse operation of roll-up. It is performed by either of the following ways:**
  - **By stepping down a concept hierarchy for a dimension.**
  - **By introducing a new dimension.**

Drill down – Example:



Drill down:

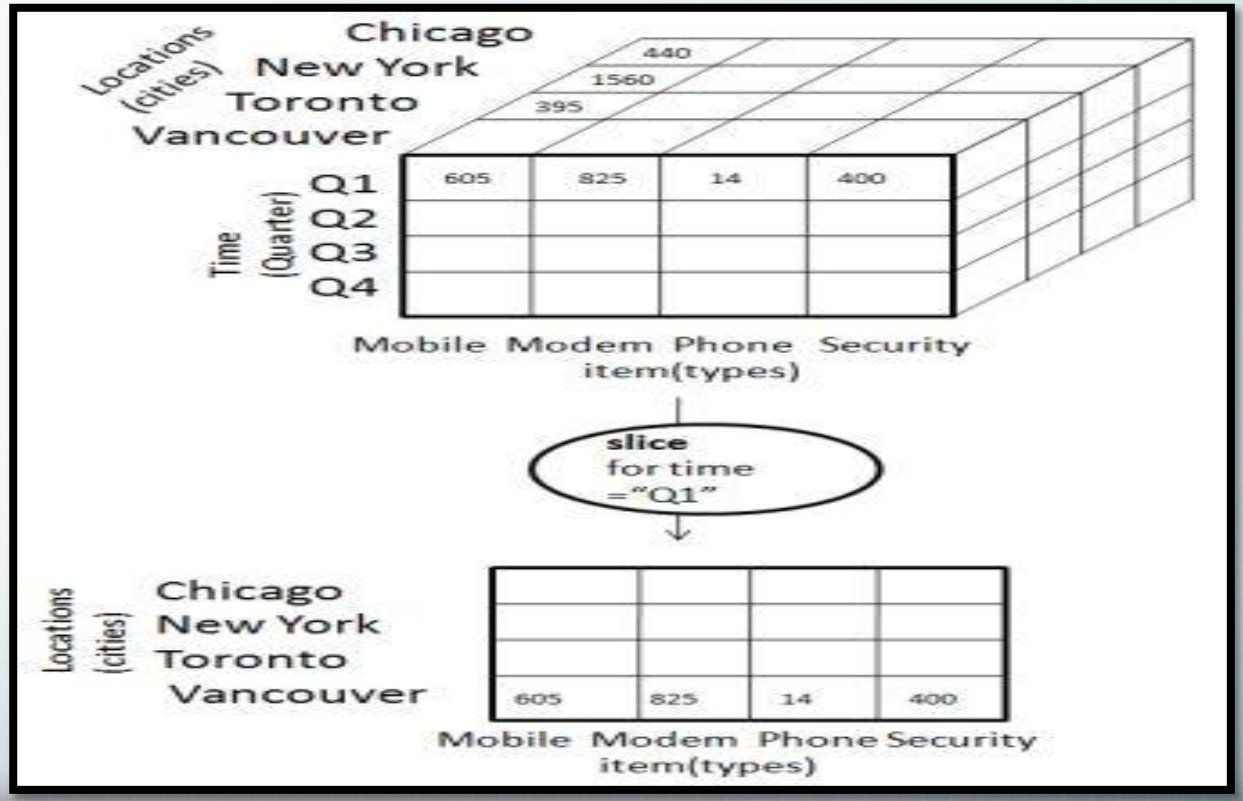
- Drill-down is performed by stepping down a concept hierarchy for the dimension time.
- Initially the concept hierarchy was "day < month < quarter < year."
- On drilling down, the time dimension is descended from the level of quarter to the level of month.

Drill down:

- When drill-down is performed, one or more dimensions from the data cube are added.
- It navigates the data from less detailed data to highly detailed data.

Slice

- **The slice operation selects one particular dimension from a given cube and provides a new sub-cube.**

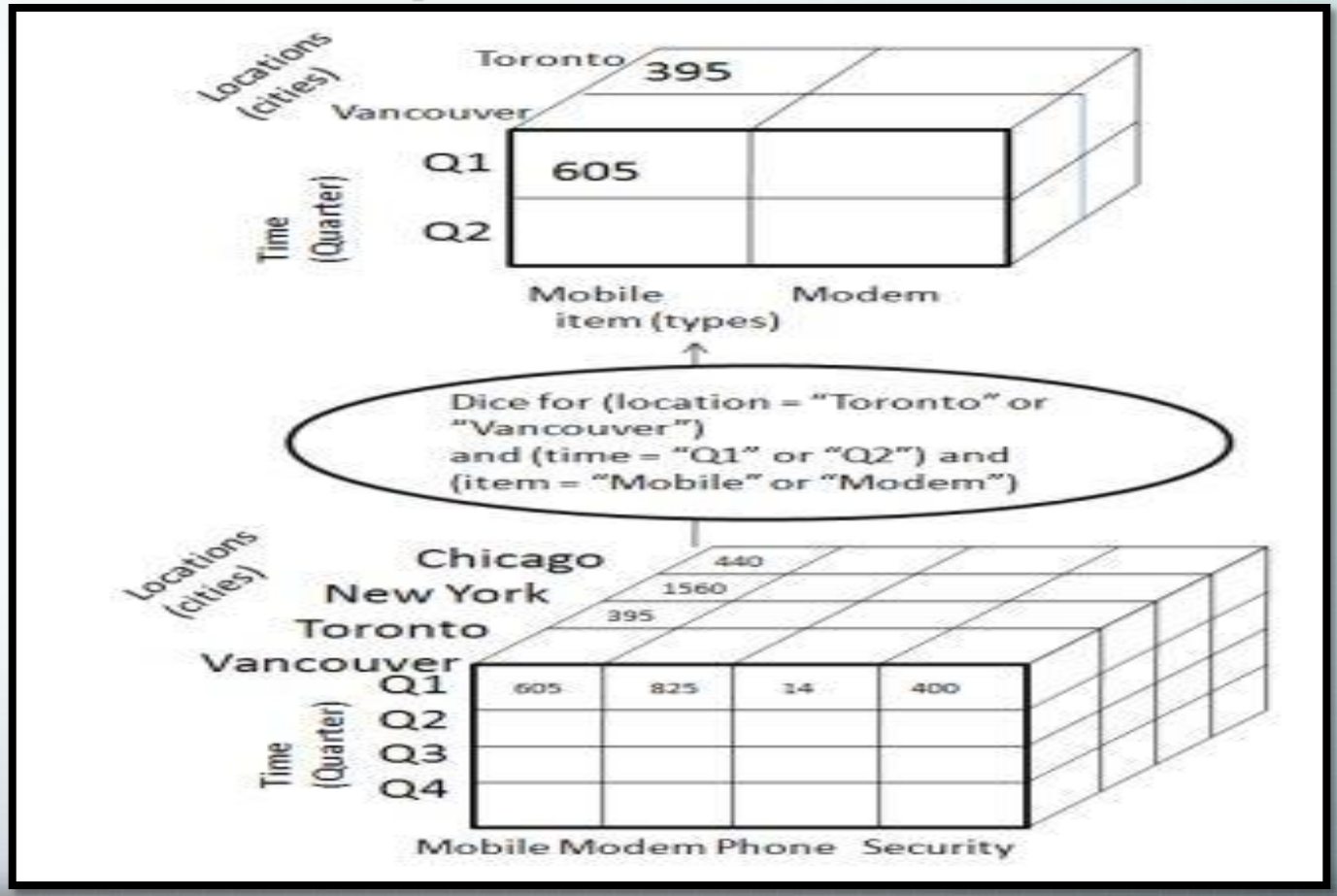


Slice Operation

- **Here Slice is performed for the dimension "time" using the criterion time = "Q1".**
- **It will form a new sub-cube by selecting one or more dimensions.**

Dice

- **Dice selects two or more dimensions from a given cube and provides a new sub-cube.**



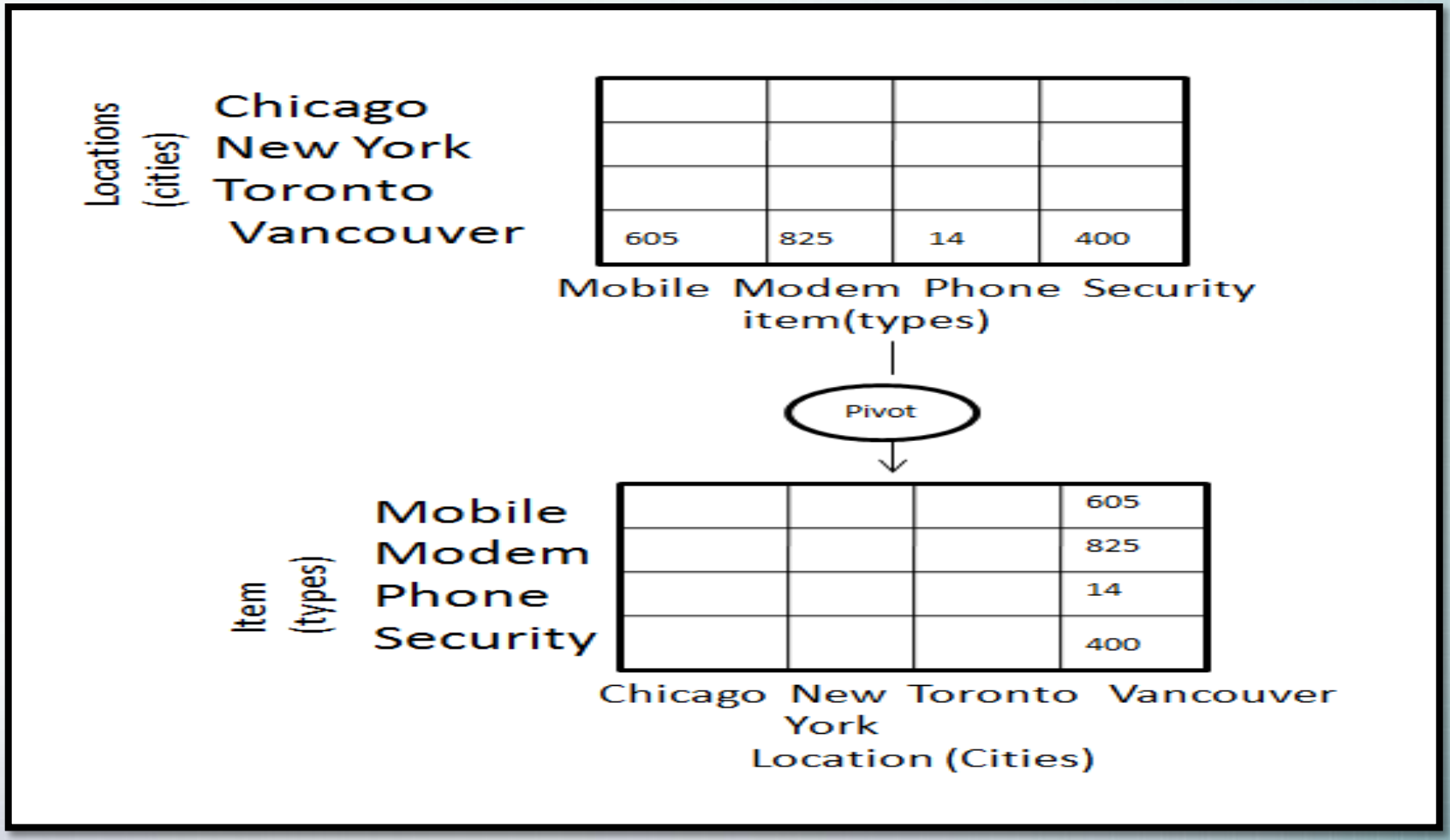
Dice Operation

- **The dice operation on the cube based on the following selection criteria involves three dimensions.**
  - **(location = "Toronto" or "Vancouver")**
  - **(time = "Q1" or "Q2")**
  - **(item =" Mobile" or "Modem")**

Pivot

- The pivot operation is also known as rotation.
- It rotates the data axes in view in order to provide an alternative presentation of data.
- In this below diagram , the item and location axes in 2-D slice are rotated.

Pivot - Example:



Denormalization

Denormalization - Definition

- ***Denormalization*** **is a strategy that database managers use to increase the performance of a database infrastructure.**

- **It involves *adding redundant data to a normalized database* to reduce certain types of problems with database queries that combine data from various tables into a single table.**

- **The definition of Denormalization is dependent on the definition of normalization, which is defined as the process of organizing a database into tables correctly to promote a given use.**

Explanation

- In many cases, denormalization involves creating separate tables or structures so that queries on one piece of information will not affect any other information tied to it.

- Therefore, database handlers will separate the two pieces of information, sometimes with redundant data, so that they can be worked on separately.

Explanation

- Where denormalization comes in is that adding redundant data allows for more sophisticated search results.
- Some examples that are typically given to explain this include situations where database handlers want to find purchase histories, or anything else about a customer or client that doesn't address the specific present state of that account.
- This is where having redundant data can allow databases to give different results based on exactly what the user is asking for.

Explanation

- **Again, having this redundant data can also improve performance based on the specific ways that a database searches for a particular item.**

- ***Challenges involved in denormalization*** **:**

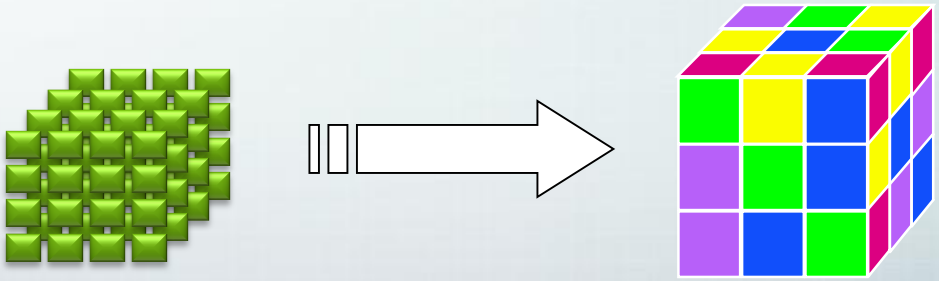
- **Incudes' documenting the process carefully to avoid some kinds of anomalies that can occur as a result of data mismatch.**

SLOWLY CHANGING DIMENSION's

WHAT IS A SLOWLY CHANGING DIMENSION?

- **Although dimension tables are typically static lists, most dimension tables do change over time.**

- **Since these changes are smaller in magnitude compared to changes in fact tables, these dimensions are known as slowly growing or slowly changing dimensions.**



SLOWLY CHANGING DIMENSION's - CLASSIFICATION

- **Slowly changing dimensions are classified into three different types as follows:**
  - **TYPE I**
  - **TYPE II**
  - **TYPE III**

SLOWLY CHANGING DIMENSIONS TYPE-I

Source

Emp id	Name	Email
1001	Shane	Shane@ xyz.com

Target

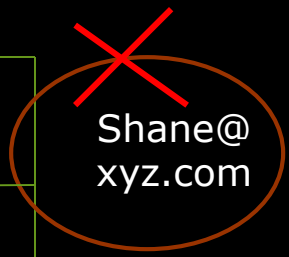
Emp id	Name	Email
1001	Shane	Shane@ xyz.com

Source

Emp id	Name	Email
1001	Shane	Shane@ abc.co.in

Target

Emp id	Name	Email
1001	Shane	Shane@ abc.co.in



SLOWLY CHANGING DIMENSIONS TYPE-II

Source

Emp id	Name	Email
1001	Shane	Shane @xyz. com

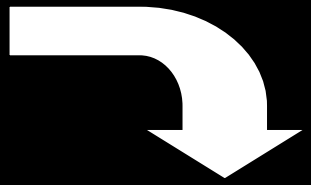
Target

PM_ PRIMARY KEY	Emp id	Name	Email	PM_ VERSION _NUMBER
1	1001	Shane	Shane @xyz. com	0

SLOWLY CHANGING DIMENSIONS -VERSIONING

Source

Emp id	Name	Email
1001	Shane	Shane@ abc.co.in



PM_PRIMARYKEY	Emp id	Name	Email	PM_VERSION_NUMBER
1	1001	Shane	Shane@xyz.com	0
2	1001	Shane	Shane@abc.co.in	1

Target

SLOWLY CHANGING DIMENSIONS –VERSIONING

Emp id	Name	Email
1001	Shane	Shane@ abc.com

Source

Target

PM_PRIMARYKEY	Emp id	Name	Email	PM_VERSION_NUMBER
1	1001	Shane	Shane@ xyz.com	0
2	1001	Shane	Shane@ abc.co.in	1
3	1001	Shane	Shane@ abc.com	2

SLOWLY CHANGING DIMENSIONS TYPE II – FLAG

Emp id	Name	Email
1001	Shane	Shane @xyz. com

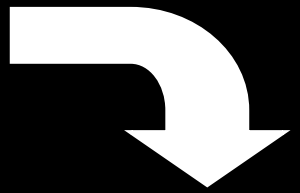
Source

PM_ PRIMARY_ KEY	Emp id	Name	Email	PM_ CURENT _FLAG
1	1001	Shane	Shane @xyz .com	1

Target

SLOWLY CHANGING DIMENSION – CURRENT FLAG

Emp id	Name	Email
1001	Shane	Shane@ abc.co.in



Source

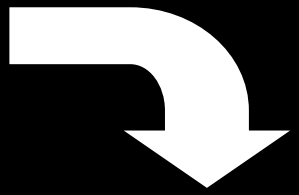
PM_PRIMARYKEY	Emp id	Name	Email	PM_CURRENT_FLAG
1	1001	Shane	Shane@ xyz.com	**0**
2	1001	Shane	Shane@ abc.co.in	**1**

Target

SLOWLY CHANGING DIMENSION - CURRENT FLAG

Emp id	Name	Email
1001	Shane	Shane@ abc.com

Source



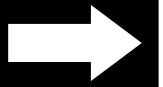
Target

PM_PRIMARYKEY	Emp id	Name	Email	PM_CURRENT_FLAG
1	1001	Shane	Shane@ xyz.com	0
2	1001	Shane	Shane@ abc.co.in	0
3	1001	Shane	Shane@ abc.com	1

SLOWLY CHANGING DIMENSIONS TYPE - II – EFFECTIVE DATE

Emp id	Name	Email
1001	Shane	Shane @xyz. com

Source

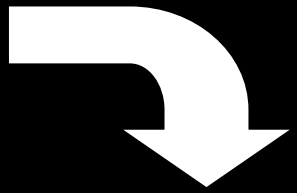


PM_PRIMARY KEY	Emp id	Name	Email	PM_BEGIN_DATE	PM_END_DATE
1	1001	Shane	Shane @xyz. com	01/01/00	

Target

SLOWLY CHANGING DIMENSION - EFFECTIVE DATE

Emp id	Name	Email
1001	Shane	Shane@ abc.co.in



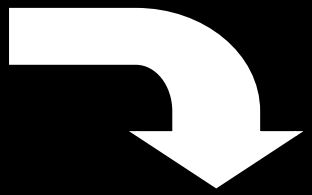
Source

PM_ PRIMARY KEY	Emp id	Name	Email	PM_ BEGIN_ DATE	PM_ END_ DATE
1	1001	Shane	Shane @xyz. com	01/01/00	03/01/00
2	1001	Shane	Shane @ abc.co. in	03/01/00	

Target

SLOWLY CHANGING DIMENSION-EFFECTIVE DATE

Emp id	Name	Email
1001	Shane	Shane@ abc.com



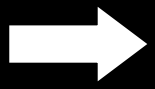
Source

Target

PM_PRIMARY KEY	Emp id	Name	Email	PM_BEGIN_DATE	PM_END_DATE
1	1001	Shane	Shane@ xyz.com	01/01/00	03/01/00
2	1001	Shane	Shane@ abc.co.in	03/01/00	05/02/00
3	1001	Shane	Shane@ abc.com	05/02/00	

SLOWLY CHANGING DIMENSIONS TYPE III

Emp id	Name	Email
1001	Shane	Shane @xyz. com



PM_PRIMARY KEY	Emp id	Name	Email	PM_Prev_Column_Name	PM_EFFECT_DATE
1	1001	Shane	Shane @xyz. com		01/01/ 00

Source

Target

SLOWLY CHANGING DIMENSIONS TYPE III

Source

Emp id	Name	Email
1001	Shane	Shane@ abc.co.in

PM_ PRIMARY_ KEY	Emp id	Name	Email	PM_Prev_ColumnName	PM_ EFFECT_ DATE
1	1001	Shane	Shane@ abc.co.in	Shane@xyz.com	01/02/00

Target

SLOWLY CHANGING DIMENSIONS TYPE III

Source

Emp id	Name	Email
1001	Shane	Shane@ abc.com

PM_PRIMARYKEY	Emp id	Name	Email	PM_Prev_ColumnName	PM_EFFECT_DATE
1	1001	Shane	Shane@ abc.com	Shane@ abc.co.in	01/03/00

Target