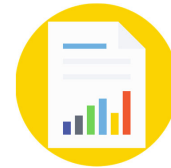


# Data Analytics Best Practices

## 3.2



## Data Warehousing Basics

(Number of Slides: 69+24)

© 2018 NUS. The contents contained in this document may not be reproduced in any form or by any means, without the written permission of NUS ISS, other than for the purpose for which it has been supplied.

## Agenda

### Day 3 (PM)

- Data Warehousing Basics
  - Data Warehousing Introduction
  - Data Modelling Essentials

# Module Objectives

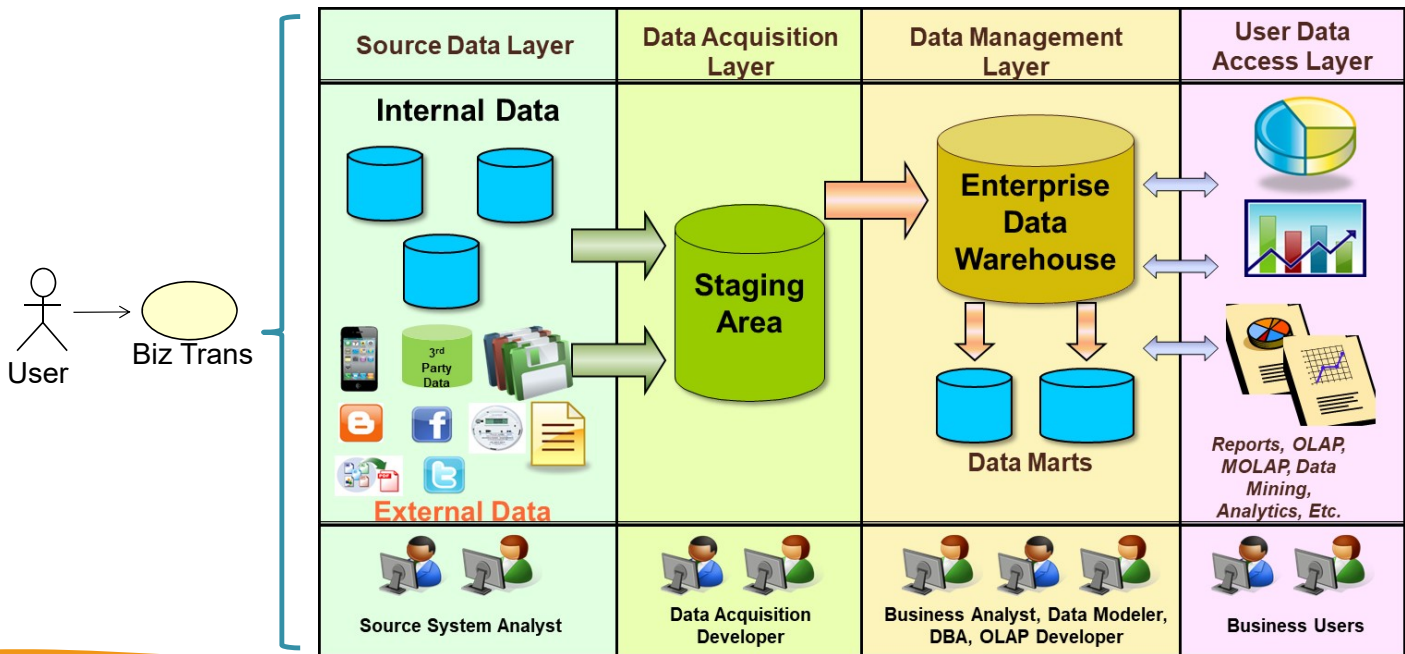
Upon completion of this Data Warehouse (DW) Essentials module, students will be able to:

- Understand the essentials & considerations required for a DW data model
- Understand the basic concepts of a DW Dimensional Model
- Understand the concepts & design considerations of the Fact and Dimension Tables

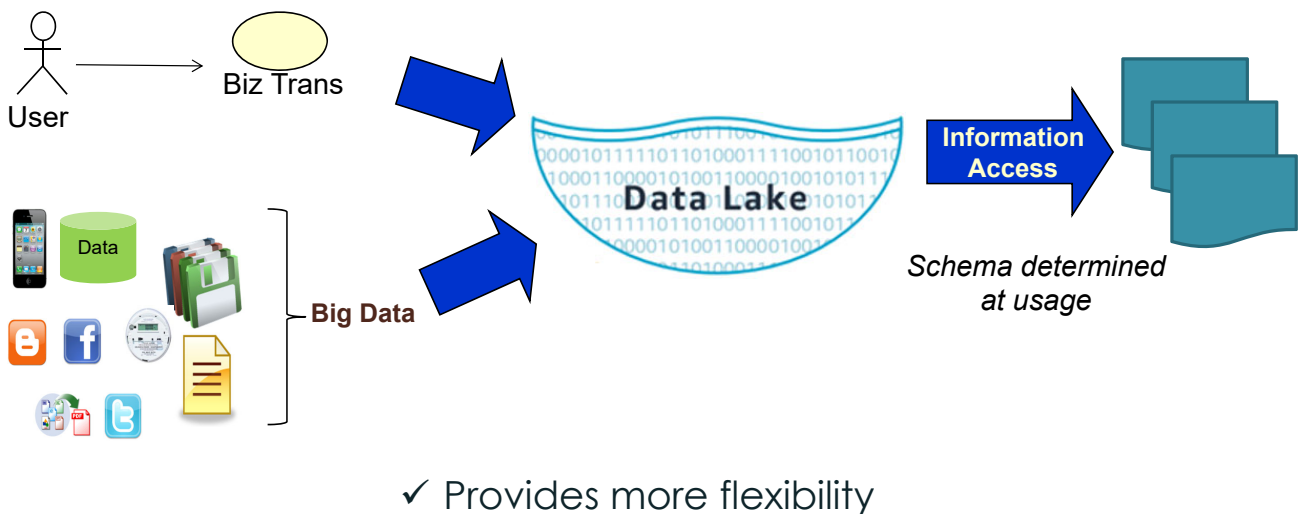
## Data Warehouse Architecture

Data Analytics Best Practices

# A Typical Data Warehouse Architecture



# Data Warehouse Architecture - A Hybrid



# Common Types of Structured Data Models

Data Analytics Best Practices

## Common Types of Structured Data Model

- Relational Model or Entity-Relational (ER) model
  - Commonly used to model RDBMS based applications
- Dimensional Model (Star Schema or Snow Flake or Constellation)
  - Popular choice in Data Warehouses
- Object Model
  - Stores rules apart from data and support inheritance
  - Popular especially in web based design

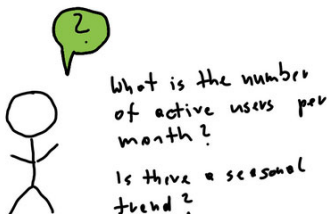
Refer to Appendix 4: Examples of Data Models

# DW Data Modeling Considerations

Data Analytics Best Practices

## The Subject Area & The Business Query...

- |                 |                  |                |                           |
|-----------------|------------------|----------------|---------------------------|
| • <b>Retail</b> | • <b>Banking</b> | • <b>Telco</b> | • <b>Manufacturing</b>    |
| – Sales         | – Customer       | – Call Usage   | – Sales                   |
| – Marketing     | – Campaign       | – Delinquency  | – Supply Chain Management |
| – Inventory     | – Deposits       | – Revenue      | – Efficiency              |



What are the Sales by **Country** by **Model** by **Day**?

What are the Sales by **Dealer** by **Model** by **Day**?



## The Subject Area (1/2)

- It is the objective of analysis & will lead to creation of the required data model (Fact & Dimension Tables)
- Work with business stakeholder to choose the “Subject Area”
- Develop the necessary business query around the selected “Subject Area”
- The “Subject Area” must be supported by data availability
- It also serves as constraints & adds richness to the user / business query

## The Subject Area (2/2)

- Start with a data exploration – check all possible data sources
- Leverage use of both “Bottom-Up” & “Top-Down” approach
- Identify the required data items
- Look for other data sources / items that could add to richness to the business query
- Design the required schema
- Organize the business perspectives (dimensions) and the business metrics or measures (facts)
- Do not normalize beyond 3<sup>rd</sup> third normal form (3NF)

# DW Modeling – Best Practices

Data Analytics Best Practices

## DW Modeling - Best Practices (1/6)

### 1) Remove operational data

- Remove attributes applicable only in the operational environment...for example:
  - Print Statement today flag
  - Last updated User
- Decide based on likely usage in analytical processing (OLAP)

## DW Modeling - Best Practices (2/6)

### 2) Add Time Element – helps to keep track of history

- Operational systems data have 'point-in-time' relationships e.g. Manager to Branch / Product to Supplier
- If an existing time element exist, try to use it e.g. Sales Date or Transaction date
- **If not, add an explicit time element**
- Many possible time element levels – for example, extraction date & time
- Time element becomes the primary key or can be concatenated to become the primary key

## Many Different Timestamp Approaches...


- 1) Single timestamp – use concept of Start time or Effective time
  - 👉 Easy to create timestamp data
  - 👎 Difficult to identify current value – requires additional field(s) to indicate current value record
  - 👎 Very difficult to identify value at a specific time period
- 2) Dual timestamp – use concept of Start time and End time
  - Helps to organize data & information into specific categories
- 3) Triple timestamp
  - It is required when operational system keys are reused
  - A time stamp must be added to indicate the start of usage of the key for one occurrence
  - 👎 Difficult to create timestamp data
  - 👉 Easy to locate identify current value and value at any time period



## Tracking Time at Attribute Level

- Apply timestamp for each attribute
- Lowest level at which changes take place
- Most detailed representation of history

Customer ID	Name	Time	Annual Income	Time	Residence Type	Time
-------------	------	------	---------------	------	----------------	------




- Unlikely that business need to keep data at this level
- Will generate large volume of time element data records
- High storage requirement, therefore not used often

## Tracking Time at Record Level

- Track attribute changes at record level
- Most commonly used for near static or “seldom change” type of data

Customer ID	Snapshot Time	Name	Annual Income	Residence Type
-------------	---------------	------	---------------	----------------



## Tracking Time at Entity (Table) Level

- Used to represent the snapshot of business as at a specific time period
- Tracks when any attribute of the record in the table changes

**CUSTOMER\_2014** 

Customer ID	Sales	Discount	Commission	Net Amt Due
-------------	-------	----------	------------	-------------

## DW Modeling - Best Practices (3/6)

### 3) Add Derived Data

- Enhance performance & reduce complexity of the reporting programs
- Recommend to store derived data for frequently accessed data items. Examples might include:
  - Sales Rev \$ = (Sales Price X Quantity)
  - Array of data = Jan Sales, Feb Sales etc.
  - SGD equivalent derived from Foreign Currency Balance
  - Gross Profit = (Revenue – Cost)
  - Number of days to maturity

## DW Modeling - Best Practices (4/6)

### 4) Separate attributes based on stability

- Group attributes based on propensity to change (Slowly Changing Dimensions) – divide into separate tables
  - Seldom change
  - Change once-a-while
  - Change often
- Tend to reduce storage requirements and lead to efficient database storage

### Stability of Customer Attributes - Example

- |                   |   |                       |
|-------------------|---|-----------------------|
| • Gender          | } | Never / Seldom change |
| • Date of Birth   |   |                       |
| • Annual Salary   | } | Changes once-a-while  |
| • Age Range       |   |                       |
| • Average Balance | } | Changes often         |
| • Customer Status |   |                       |

## DW Modeling - Best Practices (5/6)

### 5) Define the Required Granularity Levels

- Granularity is the level at which data is stored
  - Operational atomic data granularity is at the transaction level
  - DW atomic data granularity tend to be at snapshot levels
- Normally constrained at time level – time of snapshot
- More granular means more detailed data
- Lowest granularity ensures most in-depth analysis
  - For example, Sales per **product** per **store** per **day**

## Considerations for Multiple Granularity

- Performance Considerations
  - Creation of summary tables of lower granularity
  - Serves as rollup tables with pre-aggregated data
  - For example:
    - Sales by **Dealer** by **Model** by **Day**
    - Sales by **Country** by **Model** by **Day**
- Diminishing Detail Requirements
  - Detailed data value diminishes with time
  - For example:
    - Sales by **Dealer** by **Model** by **Day** for 3 months
    - Sales by **Dealer** by **Model** by **Month** for 3 years
    - Sales by **Dealer** by **Model** by **Year** for 10 years

## DW Modeling - Best Practices (6/6)

### 6) Performance enhancement - might include use of:

- Summary tables
  - Could be derived or aggregated data
- Surrogate keys
  - a unique identifier of an object or entity and is not derived from any other data in the database and may or may not be used as the primary key
- Indexes
  - Helps improves the speed of data retrieval at the cost of additional writes and storage space to maintain the index data structure

## DW Modeling - Best Practices (6/6)

### 6) Performance enhancement - might include use of: (cont'd)

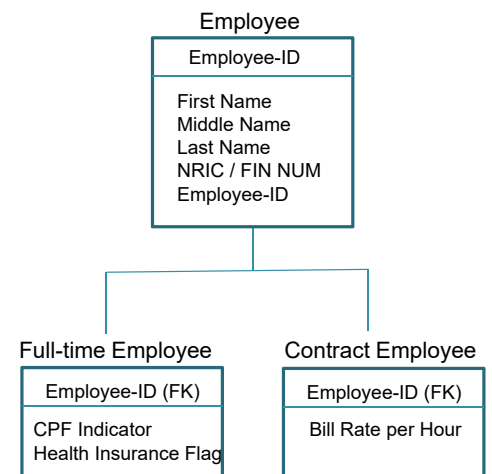
- Partitioning
  - It is a database process where very large tables are divided into multiple smaller parts so that queries that access only a fraction of the data can run faster because there is less data to scan
- Array formats
  - To store fixed-size elements of the same data type
  - Normally used to maintain relationship between child (store in array format) entity with parent entity to improve data access performance
  - For example: Invoice Header and Invoice PO Line Items

## DW Modeling – Best Practices (2/3)

**6) Performance enhancement** - might include use of: (cont'd)

- Super-type & Sub-types

- an entity type that has got relationship (parent to child) with one or more subtypes
- Contains attributes that are common to its sub-types
- Sub-types - are subgroups of the subtype entity and have unique attributes but they will be different from each sub-type
- Consider combining super-type and sub-type entities (with minimal differences)



## Dimensional Model - Introduction

Data Analytics Best Practices

## Dimensional Model – Some Terminology

- Fact table

- Central table containing “facts”
- These are business measures or metrics
- Are normally numeric

- Dimension table

- Store attributes that describe business perspectives
- They are near static data & are connected to the Fact table
- Normally include attributes dimension keys & values

- Grain / Granularity

- The level at which granular data is kept in the Fact table

- Additivity Concept

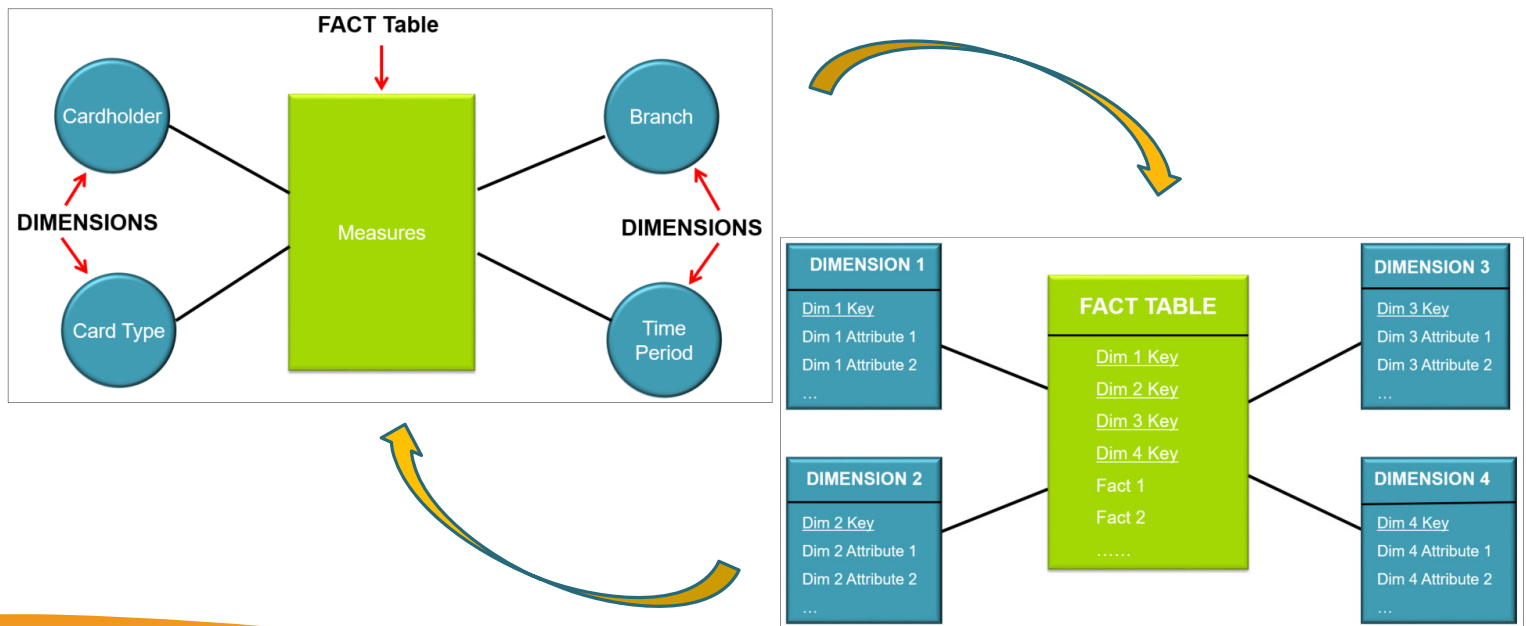
- Additive - facts that can be manipulated / added across dimensions
- Semi-additive
- Non-additive

## Dimensional Model – Basic Concepts

- Less rigorous when compared to entity/relation modeling
- Allows the designer with practical discretion to organizing tables to accommodate database complexity and to improve performance
- Leverages use of the star schema
- It represents the multi-dimensional business views using:
  - Business perspectives (dimensions)
  - Business measures (facts)
- Primarily used to model:
  - Relational Database (RDMS) for Relational Online Analytical Processing (ROLAP)
  - Multi-dimensional Database (MDB) for Multi-dimensional Online Analytical Processing (MOLAP)

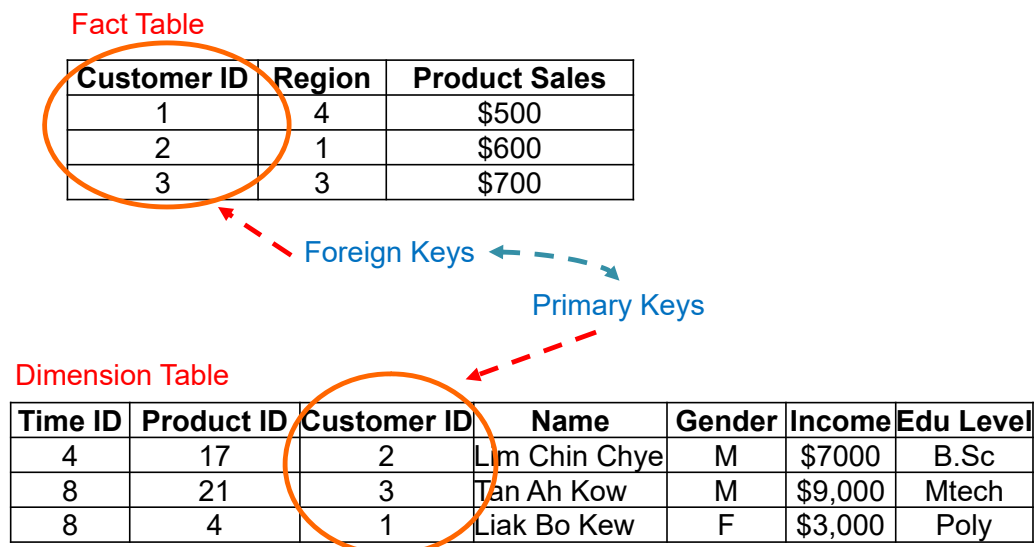
Source: Kimball Group

## The Star Schema - Example



Refer to Appendix 5: Dimensional Model Variants

## Fact & Dimension Relationship - Example





# The Fact Table

Data Analytics Best Practices

## The Fact Table - Basic Concepts (1/2)

- It is the central table and is normally de-normalized
- A direct co-relation with the identified “Subject Area”
- Time is an explicit entity of the Fact table
- **Business measures** are identified from the “Subject Area” and always have a point-in-time context
- Each row represents a single transaction
- **Most** of the data tend to be granular, numeric, continuously valued & additive in nature (e.g. Sales \$) – therefore it can be easily manipulated by aggregating or summarizing them

## The Fact Table – Basic Concepts (2/2)

- A fact table will normally consist of two categories:
  - The foreign keys category “joins” with dimension tables
  - The business measures category contain the quantitative data (information for a particular business process) that will be analyzed
- Define the granularity or atomic level of data to be stored
  - Will affect database size because lower granularity = more records
  - Fact tables are larger compared to dimensions
  - Granularity examples include:
    - Sales **per** Store **per** Product **per** Day
    - Deposit summary **per** Account at end of the Month
    - Sales **per** Product **per** Customer **per** Salesperson **per** Month
    - Payments **per** Patient **per** Month **per** Hospital

## Concept of Additivity

- Fact tables contain business measures (metrics)
- Most facts tend to be numeric, continuously valued and additive (e.g. Sales \$)
- Almost always the object of query
- Arithmetic operation are performed on metrics in queries
- Examples
  - Sales \$, Inventory, Interest Income, Budget

## Different Types of Additivity (1/3)

### (1) Additive:

- These are fact measures which can be aggregated across all dimension and their hierarchy
- Metrics (units, amounts) that can be added in any direction
- For example:
  - total sales for item over time, total sales for all items in a day

## Different Types of Additivity (2/3)

### (2) Semi-Additive:

- These are measures which can be aggregated across some of the dimensions in the fact table and their hierarchy
- Metrics (counts, averages) may not be added arbitrarily across rows and columns. For example:
  - Customer count
  - Cannot aggregate stock sales across time dimension to obtain the current stock level

## Different Types of Additivity (3/3)

### (3) Non-Additive:

- It is a specific class of fact measures which cannot be aggregated across all / any dimension in the fact table and their hierarchies
- May not be added across rows and columns in any way
- For example:
  - Aggregation of ratios; percentage or dates
  - Gross Margin %

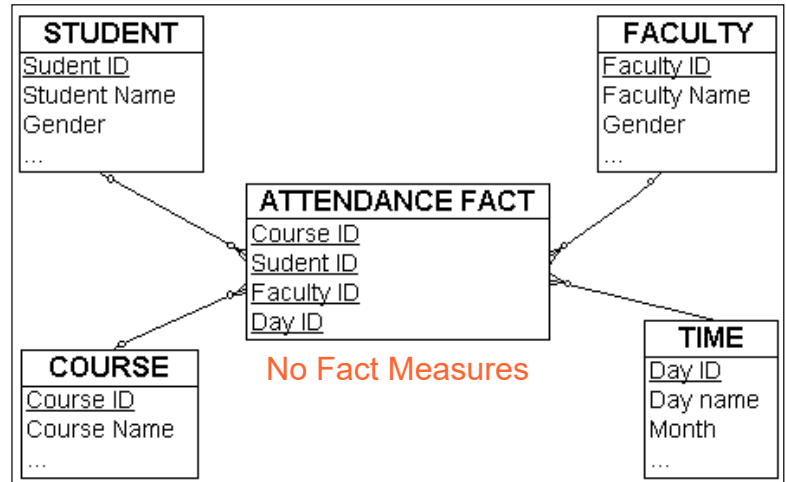
## Fact-less Fact Tables

- Dimensional design handles many-to-many relationship using fact tables
- Some event occurrences are significant with no associated fact
  - Sometimes required to analyze occurrence of an event rather than event details
- In other words - Fact table with no facts!
- Two major categories - most queries are counts of rows in the fact table
  - Event: Occurrence of an event
  - State or coverage - records inactivity

## Fact-less Fact Table for Event Tracking

Possible analytical queries include:

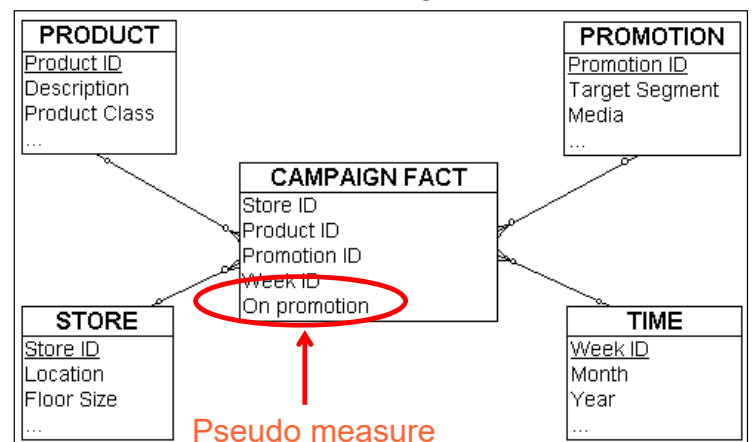
- 1) Student attendance in educational institution
- 2) What courses are most popular?
- 3) Which courses have diminishing attendance over time?
- 4) What is the influence of Gender of Faculty on attendance?



## Fact-less Fact Table for “State” (Pseudo fact)

- Inactivity can be as significant as Activity for example: Promotion Campaign Model
  - What are the promotional products over a specific time period?
  - Can be used to identify the promotional products that are not selling

### Promotion Campaign Model



## Multi-level Facts

- Also known as Grain or Granularity or Summary or Aggregation
- Often analysis is at higher levels...requires drill-down from higher level to detailed level
- Must consider keeping data at aggregate and detailed levels
- Pre-aggregation versus “on-the-fly” calculations
  - What are some considerations?

## Modeling Multi-level Facts

### Option #1

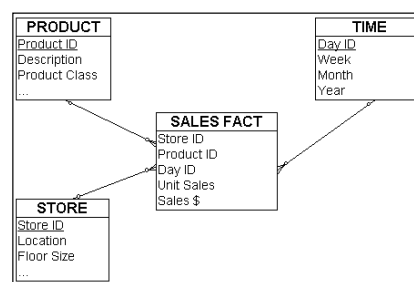
- Create multiple fact tables, one for each level of Fact

#### Pros

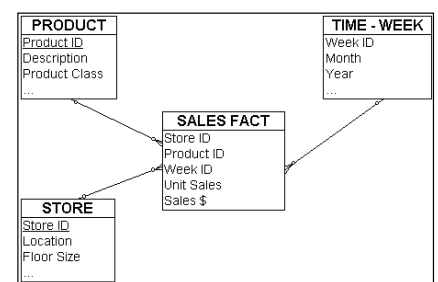
- Structure easy understand
- Many OLAP tools support summary table handling

#### Cons

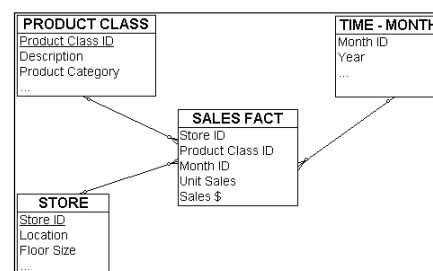
- Proliferation of Fact & Dimension tables
- Join many Fact tables to get facts at different level



Atomic level Facts



Weekly Summary



Monthly Summary

# Modeling Multi-level Facts

## Option #2

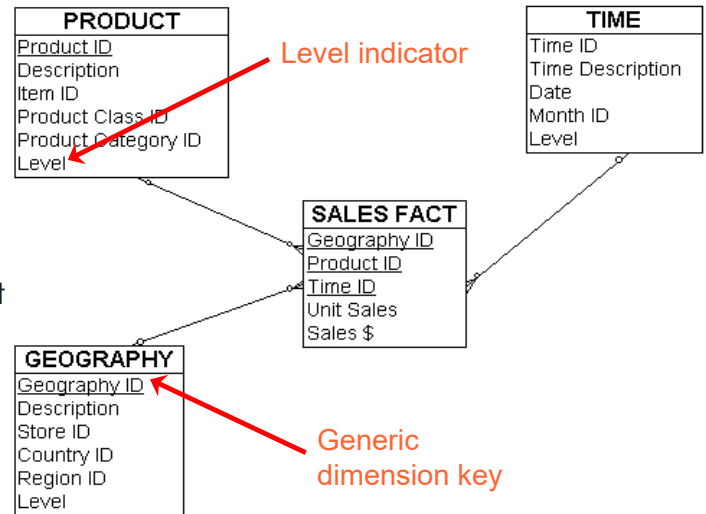
- Dimension tables
  - Generic dimension keys; Generic description column
  - Optional descriptive attributes; Level field
- Fact table
  - Only one fact table
  - Indication of level from Dimension table
  - Suitable for multi level fact availability requirement

### Pros

- Less number of tables
- Get facts of different levels from one table

### Cons

- OLAP tools limitation; Difficult to visualize
- Missing level in queries may result in invalid data returned



# Capacity Planning

- Work with your enterprise professionals
- Must decide the hardware configuration
- Must plan for the future and success of the iteration
- Must project a capacity plan for 1 to 3 years
- Make suitable assumptions on other infrastructure overheads

## Fact Table – Sizing Considerations

- For each Dimension project the number of records
- For Time Dimension, vary from 1 to 3 years to get timeline projection
- Approximate the projected number of records of each Dimension
- Approximate Fact table record length
- **Sparsity Factor** – describes % of cells in the database table that are not filled or populated with data
- **Density Factor** – describes % of cells in the database table that are filled or populated with data
- Include potential summary table requirements (2 times of fact table)
- Include database overheads (2 times of fact table)
- Include projected compounded growth rate for each year
- Unless the Dimension tables are huge, ignore them

## Calculate Size of Fact Table - Example

- Retail store example: **100** Stores & **5,000** Products
- Fact table records for **Year 1** =  $100 \times 5,000 \times 365$
- 90% Density Factor i.e. **4,500** Products sold on an average in each day in every Store
- Fact table records (adjusted) =  $100 \times 5,000 \times 365 \times \underline{90\%}$
- Approximate record length of Fact table = **64** bytes
- Fact table size (Year 1 adjusted) =  $100 \times 5,000 \times 365 \times 90\% \times 64 =$   
**10.51 GB** (approximately)



## Database Sizing for Fact Table - Example

	Y1	Y2	Y3
Basic fact table size	10.51	11.56	12.72
Summary table overheads (2x)	21.02	23.13	25.44
Database overheads (2x)	21.02	23.13	25.44
Total	52.56	57.82	63.60

Note:

- 1) All figures are in GB
- 2) Assuming compounded business growth of **10%** per annum

## Fact Design Template

<b>Attribute Name</b>	Sales \$
<b>Business Description</b>	The Sale value of a transaction, excluding GST
<b>Data type</b>	Number
<b>Length</b>	16,2
<b>Domain</b>	N.A.
<b>Mandatory ?</b>	Yes
<b>Key ?</b>	N.A.
<b>Additive ?</b>	Additive - Fully additive across all dimensions
<b>Source</b>	POS System

- Possible values for Key
  - PK-Primary Key
  - FK-Foreign or alternate or index
- Possible Values for Additive
  - Additive - Fully additive across all dimensions
  - Semi-additive - Additive across certain dimensions
  - Non-Additive - Not additive across any dimensions

## The Fact Table Conclusion ...

- Fact tables contain the business measures
- Fact tables can exist without a Fact!
- Multiple granularity Fact tables are often required
- Capacity planning for Fact tables is critical as their sizes are substantial

## The Dimension Table

Data Analytics Best Practices

## The Dimension Table - Basic Concepts

- A dimension table stores dimensions – business perspectives
- **Business perspectives** are derived from the identified “Subject Area” – they have a direct relationship
- In the retail organization example, dimension tables may include details of the customers, employees, items, stores or other objects
- For example, the items dimension table would contain a record for each item sold in the store, for example:
  - cost of the item, the supplier
  - the color, sizes and other relevant data

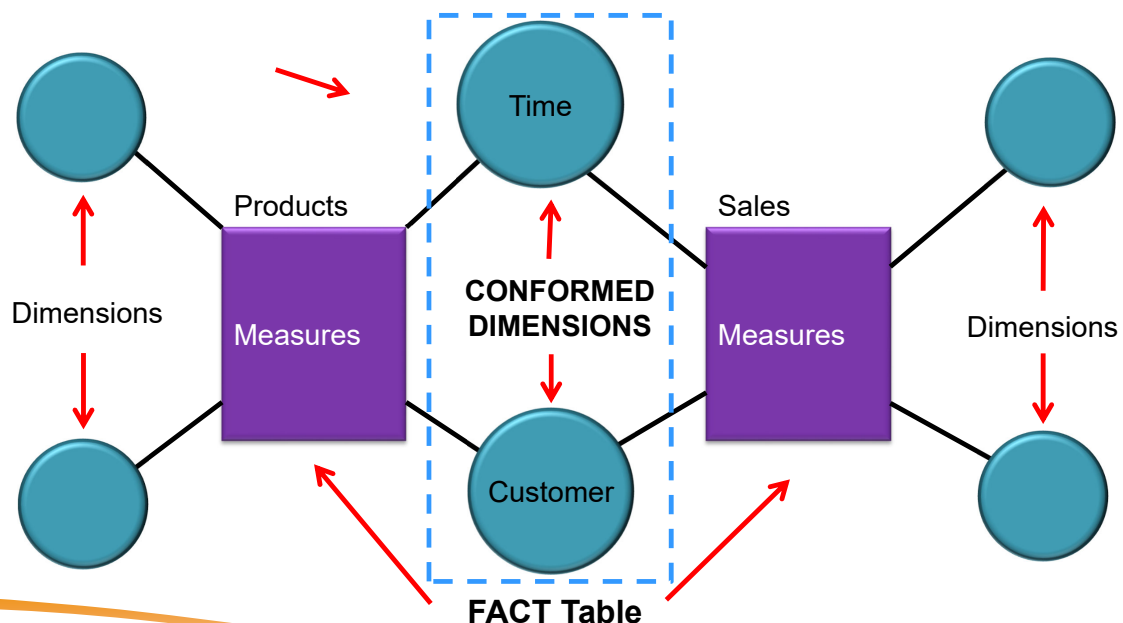
## Conformed Dimensions

Data Analytics Best Practices

## Conformed Dimensions - Basic Concepts

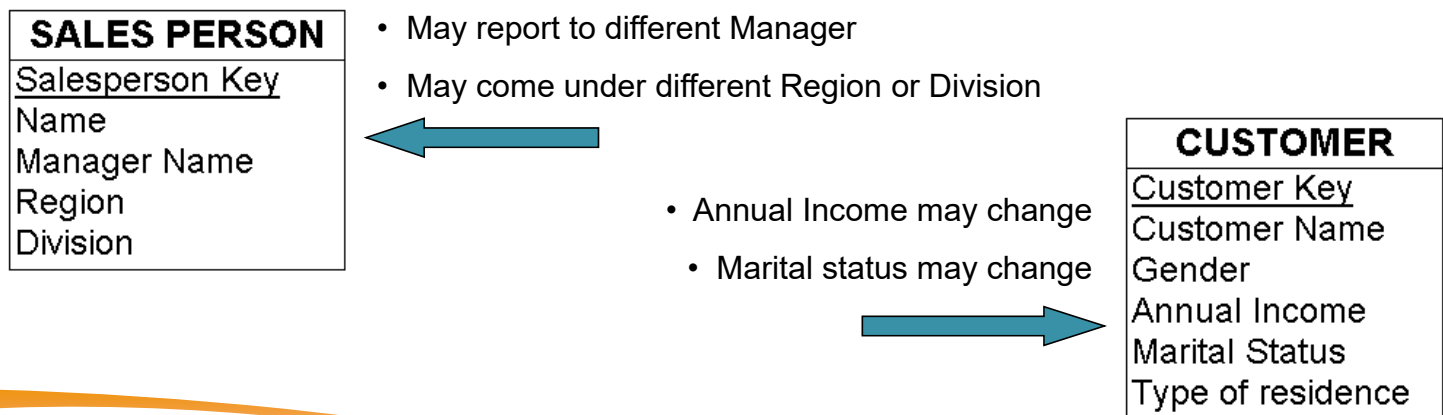
- A single, coherent view of the same piece of data throughout the data warehouse
  - One conformed dimension across multiple Fact tables
  - Provides the same meaning to all Fact tables because its data attributes have the same structure, attributes, domain values and definitions
  - Allows a consistent way to access Facts and Measures across multiple Facts → fast access and consistent reporting (single version of truth)
  - Facilitates intuitive “drill down” capabilities of data from multiple operational systems
- May have duplications across data marts

## Conformed Dimensions - Example



## Slowly Changing Dimensions (SCD)

- Slowly Changing Dimensions – their attributes change slowly over time
- The history of these attribute values may be required for trend and historical analysis



## Common Types of SCD (1/3)

- Type 1- Overwrite Previous Attribute Values

Before the Change					
Customer Key	Customer Name	Gender	Annual Income	Marital Status	Residence Type
1234567	Tan Ah Kow	Male	25000	Single	HDB - rental

After the Change					
Customer Key	Customer Name	Gender	Annual Income	Marital Status	Residence Type
1234567	Tan Ah Kow	Male	25000	Married	HDB - owned

### Pros

- Easy to implement
- Can use for insignificant changes
- Correct erroneous data

### Cons

- No history is maintained
- Same query at different times may produce different answers

## Common Types of SCD (2/3)

- Type 2 - Add an Additional Dimension Record
  - Often used option as complete history is held
  - Mandates creation of surrogate key
  - Effective date fields used to identify point-in-time records

Before the Change							
DW Customer Key	Customer Key	Customer Name	Gender	Annual Income	Marital Status	Residence Type	Effective Date
14099	1234567	Tan Ah Kow	Male	25000	Married	HDB - owned	1-Mar-13
After the Change							
DW Customer Key	Customer Key	Customer Name	Gender	Annual Income	Marital Status	Residence Type	Effective Date
14099	1234567	Tan Ah Kow	Male	25000	Married	HDB - owned	1-Mar-13
15000	1234567	Tan Ah Kow	Male	50000	Married	HDB - owned	1-Mar-14

### Pros

- Complete history kept
- Any number of changes can be tracked

### Cons

- May result in explosion of dimension records
- Query tools may treat version record as a new occurrence

## Common Types of SCD (3/3)

- Type 3 - Add New Field(s)
- Used for limited history tracking where the propensity to change is low
- Not recommended for frequently changing attributes
- The number of previous value fields can vary

Before the Change									
DW Cust Key	Cust Key	Cust Name	Gender	Annual Inc	Curr Marital Status	Curr Effective Date	Previous Martial Status	Previous Effective Date	Residence Type
14099	1234567	Tan Ah Kow	Male	25000	Single	1-Mar-13	NULL	NULL	HDB - owned
After the Change									
DW Cust Key	Cust Key	Cust Name	Gender	Annual Inc	Curr Marital Status	Curr Effective Date	Previous Martial Status	Previous Effective Date	Residence Type
14099	1234567	Tan Ah Kow	Male	25000	Married	1-Mar-14	Single	1-Mar-13	HDB - owned

### Pros

- Lesser dimension records compared to Type 2
- Historical values are available in one record

### Cons

- Only limited history possible
- Query of point-in-time records difficult
- Record structure becomes complicated
- Trend analysis difficult

# DW Hierarchies

Data Analytics Best Practices

## What are DW Hierarchies?

- Hierarchies are logical entities or structures that are used for analysis
- Hierarchies can be used to organize data aggregation; their paths and define drill path and to establish a family structure
- May result in one or multiple levels
- Each level is connected to the levels above and below it
- This is a key advantage because query tools use hierarchies to enable drill down to view different levels of granularity

## DW Hierarchies - Design Considerations

- Must consider relationships in business structures
  - For example: divisional multi-level sales organization
- Data values at lower levels (child) aggregate into data values at higher levels (parent)
- Levels define parent child relationship
- Most general (root) to the most specific information
- For example, aggregate sales revenue on a monthly basis to a quarterly and to a yearly aggregation when dimensional dependencies between month, quarter and year are defined

## Examples of Levels and DW Hierarchies





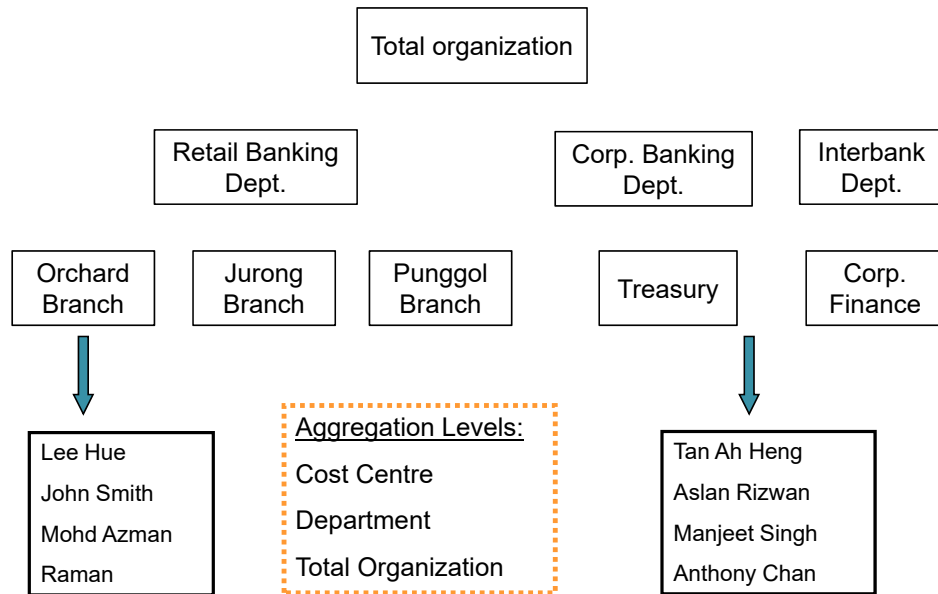
## Hierarchy Example - Bank

Organization

Department

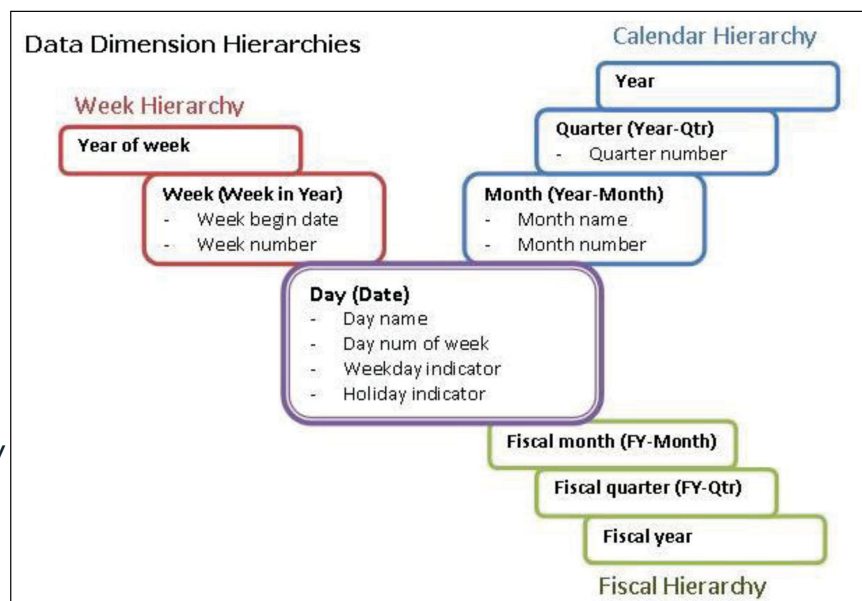
Cost Centre

Account Officer



## Alternate Hierarchy

- When multiple aggregation paths are possible for same dimension, it may give rise to 'alternate hierarchy'
- Time Dimension
  - DW Dimension Year Hierarchy
  - Calendar Year Hierarchy
  - Fiscal Year Hierarchy



## Dimension Design Template

<b>Attribute Name</b>	Store Location
<b>Business Description</b>	Geographical location of the store
<b>Data type</b>	Character
<b>Length</b>	30
<b>Domain</b>	N.A.
<b>Mandatory?</b>	Yes
<b>Volatility</b>	I - Changes infrequently
<b>History Preference</b>	PREV - Keep current and previous values
<b>Source</b>	POS System

- Possible values for Volatility
  - N - Never changes
  - I - Changes infrequently
  - F - Changes frequently

- Possible values for History Preference
  - NO - No History required
  - FULL - Full History required
  - PREV - Keep current and previous values
  - NA - Not Applicable

## The Dimension Table - Conclusion

- ✓ Hierarchies defined as Dimension attributes
- ✓ History requirement for Dimension attributes is critical in design
- ✓ Normalization concepts can be applied in a limited manner
- ✓ Dimension data volumes affect design

## Summary...

- Converting operational model to DW model is a science
- Multiple approaches to track history & to ensure performance in the DW
- Stability of attributes affect grouping into entities
- DW tends to have multiple granularity of data
- Dimensional model process – many variant exists
- Fact Tables – contains business metrics or measures, always joined to a Dimension table, much larger than a dimension table
- Dimension tables - contains data on business perspective, primary key, always joined to a Fact table (1:Many), contains one or more descriptive attributes

## Appendix 1: Data and Database Fundamentals

Data Analytics Best Practices

## Some Database Fundamentals...

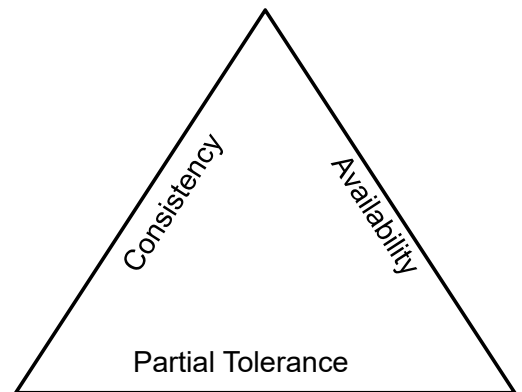
- CAP Theorem – for Distributed Systems
- ACID
- BASE

## Accessing the Data...A Unique Phenomenon

- The relational paradigm is based on proven mathematical principles
- RDBMS stores the data structure differently from the way it is programmed
- The object-oriented paradigm is based on proven software engineering principles
- In OO programming the data structure is stored in class diagrams that typically reflect in how OO programmers will program - traverse objects via their relationships

## Distributed Systems - CAP Theorem (1/2)

- Published by Eric Brewer in 2000
- Sets the basic requirements that describe any distributed system
- The outcome - if you want consistency, availability and partition tolerance, you have to settle for two out of three
- Theoretically it is impossible to have all 3 requirements met
- Deciding factor is dependent on what technology is used



## Distributed Systems - CAP Theorem (2/2)

- **Consistency** – is the system able to operates fully or not
  - Does the system reliably follow the established rules within its programming according to those defined rules?
  - Do all nodes within a cluster see all the data they are supposed to? This is the same idea presented in ACID.
- **Availability** – is the system available when required
  - Is the given service or system available when requested?
  - Does each request get a response outside of failure or success?
- **Partition Tolerance** – is the system able operate even under circumstances of data loss or system failure
  - A single node failure should not cause the entire system to collapse.

# SQL Relational Systems - ACID

- A set of properties that applies specifically to structured database transactions
  - **A**tomic - Everything in a transaction succeeds or the entire transaction is rolled back.
  - **C**onsistent - A transaction cannot leave the database in an inconsistent state.
  - **I**solated - Transactions cannot interfere with each other.
  - **D**urable - Completed transactions persist, even when servers restart etc.

## BASE <sub>(1/2)</sub>

**BASE** stands for: **B**asically **A**vailable **S**oft-state & **E**ventual consistency

- Basically Available
  - Refers to the perceived availability of the data. If a single node fails, part of the data won't be available, but the entire data layer stays operational
  - For example, waiting for a check to clear in your bank account
- Soft state
  - The state of the system is always soft
  - With or without inputs, it changes over time – with or without inputs – due to 'eventual consistency'

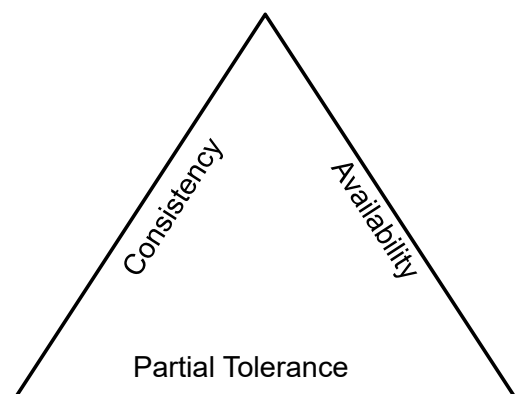
## BASE (2/2)

- Eventual consistency

- The system will not check for consistency of every transaction before it moves to the next transaction
- Since the data will propagate to all nodes, it will *eventually* become consistent once it stops receiving input
- Rather than requiring consistency after every transaction, it is enough for the database to eventually be in a consistent state
- It's OK to use stale or give approximate answers.
- Example: closing of financial systems

## Quick Comparison (1/2)

- **CAP** is a continuum - Consistency Availability & Partition Tolerance → theoretically impossible to achieve all 3
- **ACID** focuses on Consistency and Availability
- **BASE** focuses on Partition Tolerance and Availability but not Consistency



## Quick Comparison (2/2)

### Distributed Systems

- Large number of distributed servers working together
- It provides the basic requirements that a distributed storage system has to follow
- Data storage alternatives are available for data varieties
- Most NoSQL systems do not implement ACID and vary in how durable they are with stored data
- Depending on the software used or needs of the system, it will have to fit 2 of the 3 requirements of the CAP theorem

### Structured Relational Systems

- This is a set of rules that a database can choose to follow that guarantees how it handles transactions and keeps data safe
- Committed data is considered reliable

## Appendix 2: RDBMS Versus NoSQL

Data Analytics Best Practices



## RDBMS Vs NoSQL Pros

- RDBMS pros:
  - ACID transactions at the database level makes development easier.
  - Fine-grained security on columns and rows using views prevents views and changes by unauthorized users.
  - Most SQL code is portable to other SQL databases, including open source options.
  - Typed columns and constraints will validate data before it's added to the database and increase data quality.
  - Existing staff members are already familiar with entity-relational design and SQL.
- NoSQL pros:
  - Loading test data can be done with drag-and-drop tools before ER modeling is complete.
  - Modular architecture allows components to be exchanged.
  - Linear scaling takes place as new processing nodes are added to the cluster.
  - Lower operational costs are obtained by auto-sharding.
  - Integrated search functions provide high-quality ranked search results.
  - There's no need for an object-relational mapping layer.
  - It's easy to store high-variability data.

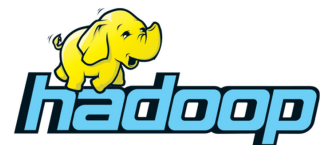
## RDBMS Vs NoSQL Cons

- RDBMS cons:
  - The object-relational mapping layer can be complex.
  - Entity-relationship modeling must be completed before testing begins, which slows development.
  - RDBMSs don't scale out when joins are required.
  - Sharding over many servers can be done but requires application code and will be operationally inefficient.
  - Full-text search requires third-party tools.
  - It can be difficult to store high-variability data in tables.
- NoSQL cons:
  - ACID transactions can be done only within a document at the database level. Other transactions must be done at the application level.
  - Document stores don't provide fine-grained security at the element level.
  - NoSQL systems are new to many staff members and additional training may be required.
  - The document store has its own proprietary nonstandard query language, which prohibits portability.
  - The document store won't work with existing reporting and OLAP tools.

# Appendix 3: NoSQL Hadoop Introduction

Data Analytics Best Practices

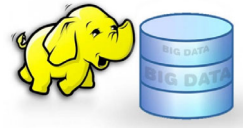
## NoSQL - Hadoop



- Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware \*\*\*
- Hadoop was inspired by Google's MapReduce, a software framework
- Hadoop's creator - Doug Cutting - named the framework after his child's stuffed toy elephant
- Basically an application is broken down into numerous small parts that can be run on any nodes in the cluster

\*\*\* Source: [https://en.wikipedia.org/wiki/Apache\\_Hadoop](https://en.wikipedia.org/wiki/Apache_Hadoop)

# NoSQL - Hadoop



- Applications on systems will run on with thousands of nodes involving thousands of terabytes
- It has a distributed file system across the nodes
- Designed to scale up from a single server to thousands of machines, with very high degree of fault tolerance
- Performance is design and programmed in the application layer
- High degree of fault tolerance - resiliency of these clusters comes from the software's ability to detect and handle failures at the application layer
- It facilitates rapid data transfer rates among nodes and allows the system to continue operating uninterrupted in case of a node failure – lowers risk of failure
- Provides better scalability, lower latency, greater flexibility, and a better price/performance ratio in an age of Big Data and Cloud computing

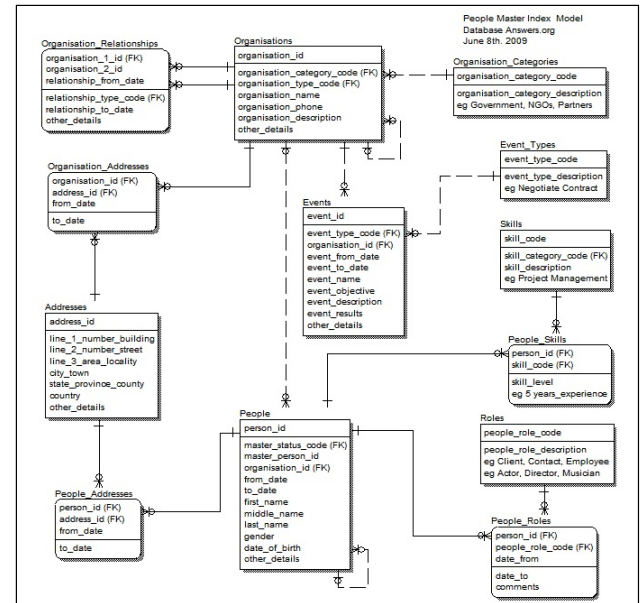
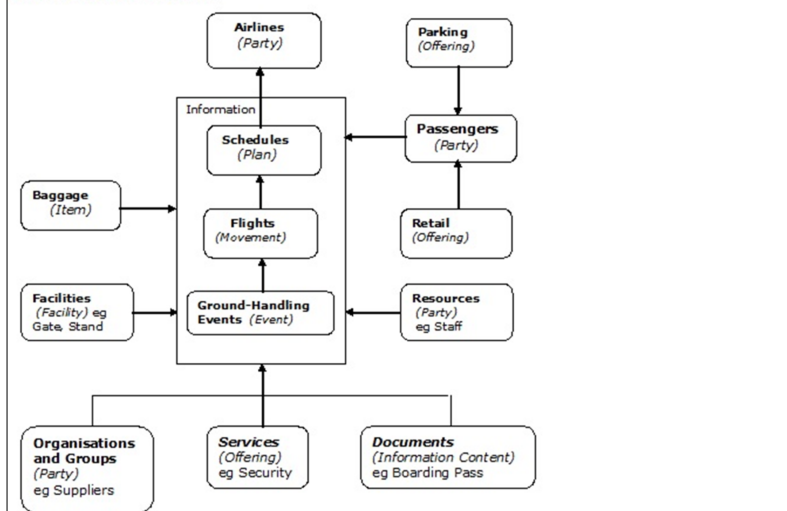
Source: Tech Target

## Appendix 4: Examples of Data Models

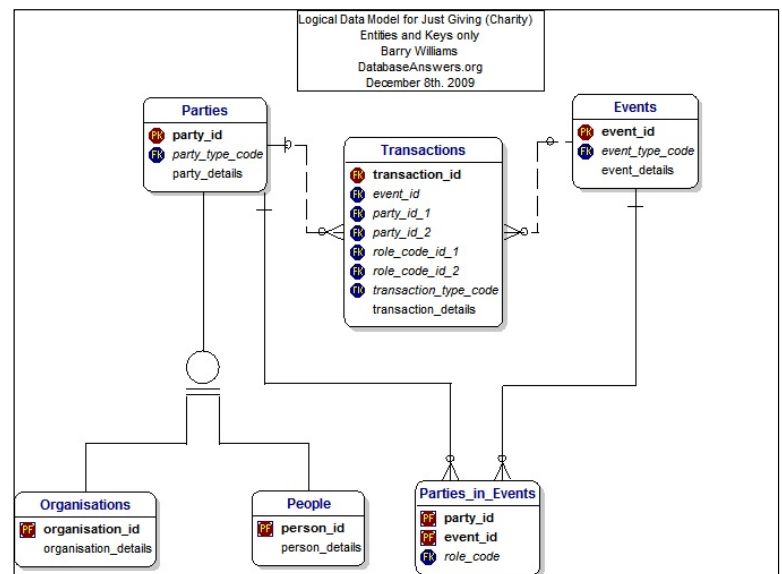
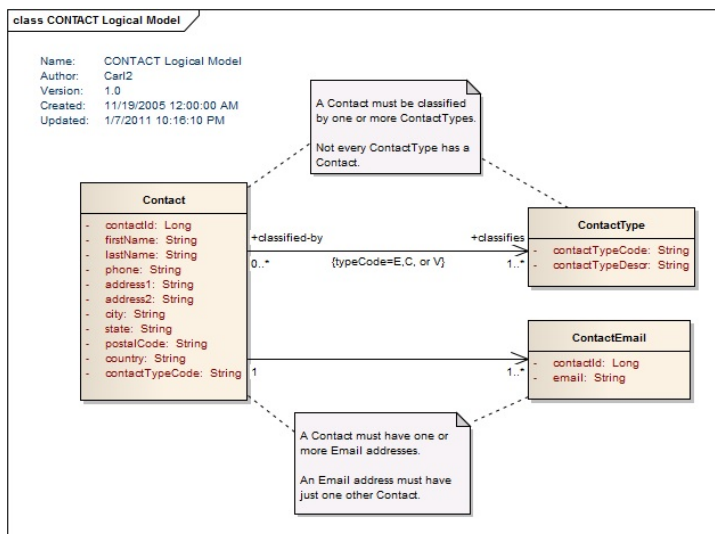
Data Analytics Best Practices

# Example - Conceptual Data Model

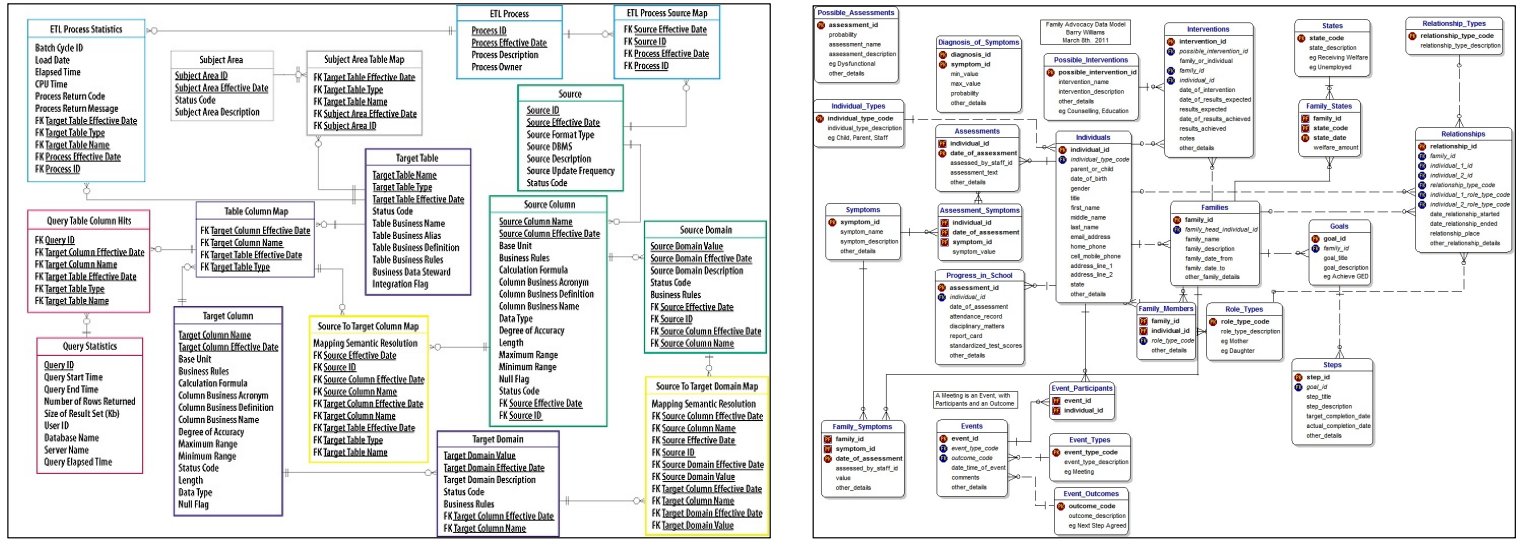
The arrows go from Children to Parents.



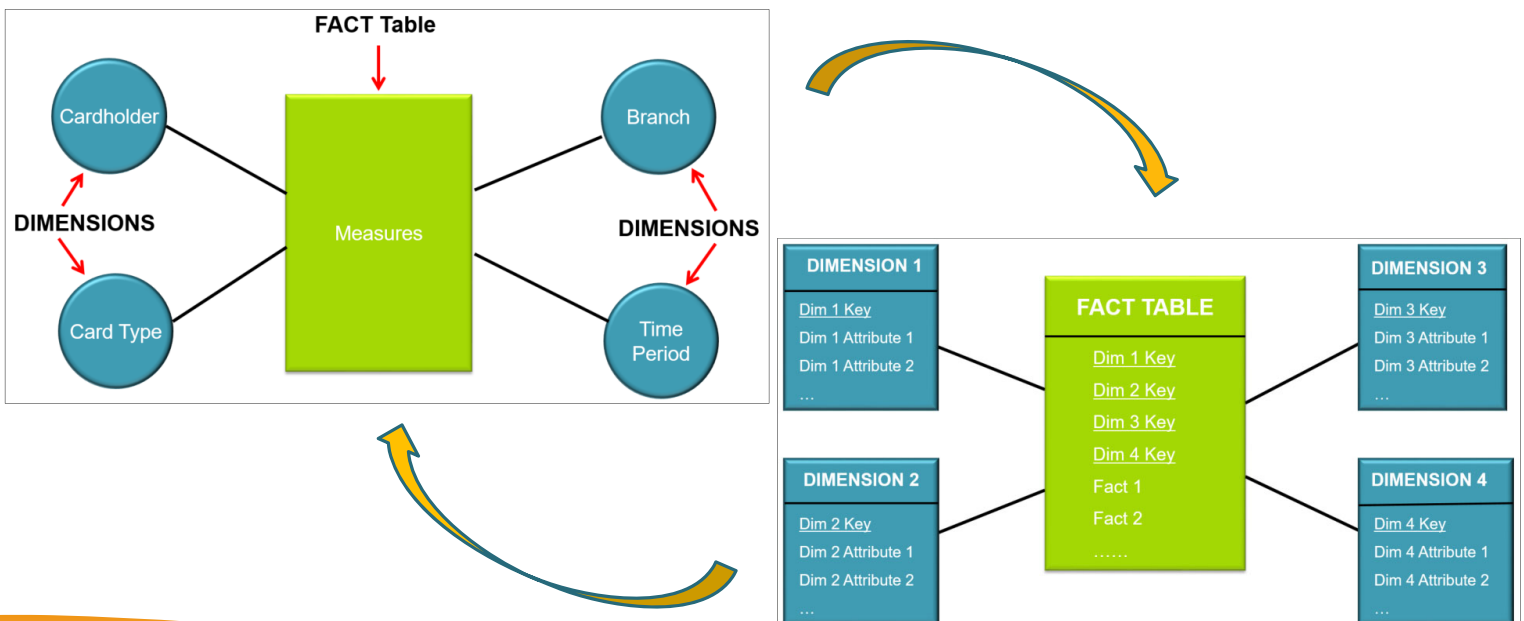
# Example – Logical Data Model



# Example – Physical Data Model



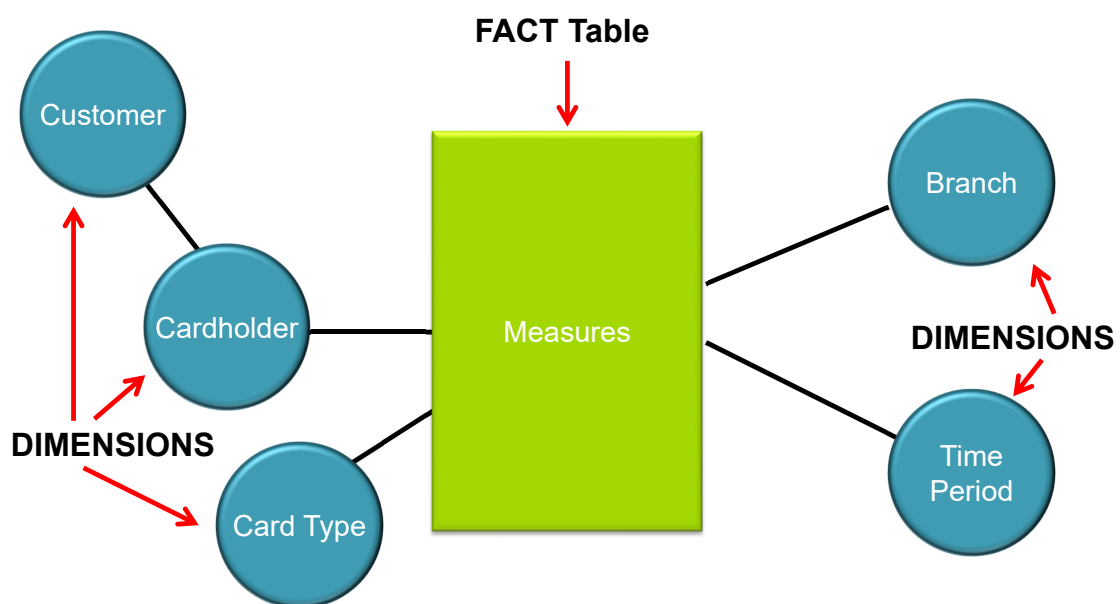
# Dimensional Model - Star Schema



# Appendix 5: Dimensional Model - Variants

Data Analytics Best Practices

## Dimensional Model Variants - Snowflake Schema



## Dimensional Model Variants - Star Constellation

