**Data Warehousing: Main note, note1**

1. Data warehousing(BI, Business Intelligence) solution is a collection of objects that allows data to be turned into useful information.

**BI Life Cycle**

1. Interview and Identify Data
   1. Determine the solution requirements and isolating the data you will be working with
2. Plan the BI solution
   1. Document the requirements, procedures
3. Create a data warehouse
   1. Design the RDBMS Database
4. Create the ETL process
   1. Extract-Transform-Load
      1. Create
      2. test ETL scripts
      3. Connections
      4. Load data into data warehouse
5. Cube creation
   1. Create cubes and determine storage solutions
      1. ROLAP
         1. Relational Online Analytical Processing
      2. HOLAP
         1. Hybrid OLAP
      3. OLAP
         1. Online Analytical processing
6. Create Reports
   1. Report against cubes and/or other sources
7. Test and fine tune the solution
8. Go Live, Approve, release and plan for updates

A close up of text on a white background

Description automatically generated

**Main Components of a Data Warehouse**

1. Operational Source Systems
   1. There are OS captures the transactions of the business(ie: OLTP). The source systems should be thought of as outside the Data Warehouse.
2. Data Staging Area
   1. Storage area and a set of processes commonly referred to as extract-transformation-load (ETL). The data staging area is everything between the operational source systems and data presentation area.
3. Data Presentation
   1. Data is organized, and made available for querying by users and analytical applications.
4. Data Access Tools

**Why do we need a Data warehouse?**

1. To perform querying and reporting on servers/disks not used by transaction processing systems. Don’t want to interface with the functioning and performance of the transactional database.
2. To use data models and/or server tech that speed up querying and reporting that are not appropriate for transaction processing
3. To provide a repo of “cleaned up” transaction processing systems data that can be reported against.
4. To make it easier to query and report data from multiple transaction processing systems and/or from external data sources and/or from data that must be stored for query/report purposes only.
5. Security? To prevent persons who only need to query and report transaction processing system data from having any access to transaction processing system db and logic used to maintain those db.

**Who uses the Data warehouse?**

1. Designed to executives, senior managers, business analysts in making complex business decisions.

**Data Warehousing Terminology**

1. Dimensional Modeling
   1. Relational DB(OLTP) normalize. but not in Data warehouse
   2. Highly normalized designs don’t benefit Data warehouse.

**Dimensional Modeling**

* Relational DB(OLTP) are designed using Entity Relationship(ER) model
* **But** Data warehouse DB are designed using Dimensional Model
* Relational model removes redundancies by normalization.
  + It benefits OLTP DB by ensuring transactions are simple and short as possible
* Business users who access the Data warehouse are more concerned with running queries that are neither simple nor short on large amounts of data.
  + So highly normalized designs don’t benefit Data warehouse
* Most queries done on Data warehouse are done ad-hoc unlike relational DB that usually use pre-programmed queries

**Schemas**

1. **Star Schema**
   1. Fact Table = single central table
      1. Surrounded by multiple tables(dimensions)
   2. Each **Star Schema** covers one business area
      1. E.g., ‘FactInternatSales’
   3. Fact Table is connected to all dimensions with foreign keys
   4. Usually, FK taken are in the Fact Table
   5. Fact Table is ALWAYS on many side of the relationship with the dimensions.
   6. Fact Table represents a multi dimensional hypercube.
2. **Shared Dimension / Conformed Dimension**
   1. Dimensions have FK relationships with multiple fact tables. Dimensions with connections to multiple fact tables.
3. **Snowflake Schema**
   1. Star Schema dimensions are de-normalized.
   2. But **Snowflake Schema** isn’t
   3. If you normalize Star Schema, you end up with a **Snowflake Schema**
   4. **Fact Table must always still be on many side of relationship.**
4. **Hybrid Schema**
   1. When you normalize only part of the design to accommodate shared dimensions

**Then which Schema is Appropriate?**

1. **Star Schema =** best option
2. **Hybrid Schema** = if you need to share a dimension with multiple dimensions.
3. **Snowflake Schema** = only use for quick proof of concept project
   1. This design is closest to the source DB that is already in 3NF.

**Granularity**

**Dimensionality**

1. Dimensional granularity of fact table
   1. dependent on number of dimensions connected to Fact Table

**Types of Tables**

1. **Fact Table** = contain all measures
   1. FK to dimension tables
   2. All non-key columns(measures), numeric and usually additive
   3. PK of Fact Table can consist of
      1. PK from dimension tables, therefore, having many FK and a composite primary key
      2. You can also create an auto number type field as PK
   4. Fact Table changes constantly(new rows are added constantly)
   5. Contains a single index based on the PK
   6. Examples of measures
      1. Sale Price
      2. Cost
      3. Quantity
2. **Dimensional Table** = describe measures
   1. Contains descriptive textual data that describes dimensions
   2. Don’t contain foreign keys(in a Star Schema)
   3. Contain only single **column** PK, make it auto number type field
   4. Should contain a surrogate key(it’s PK for dimension table)
   5. Usually small in size
   6. Usually change very slowly
   7. May contain numerous indexes based on textual data
   8. Examples of dimension tables
      1. Time, Employee, product

**How Fact Tables and Dimension Tables interact**

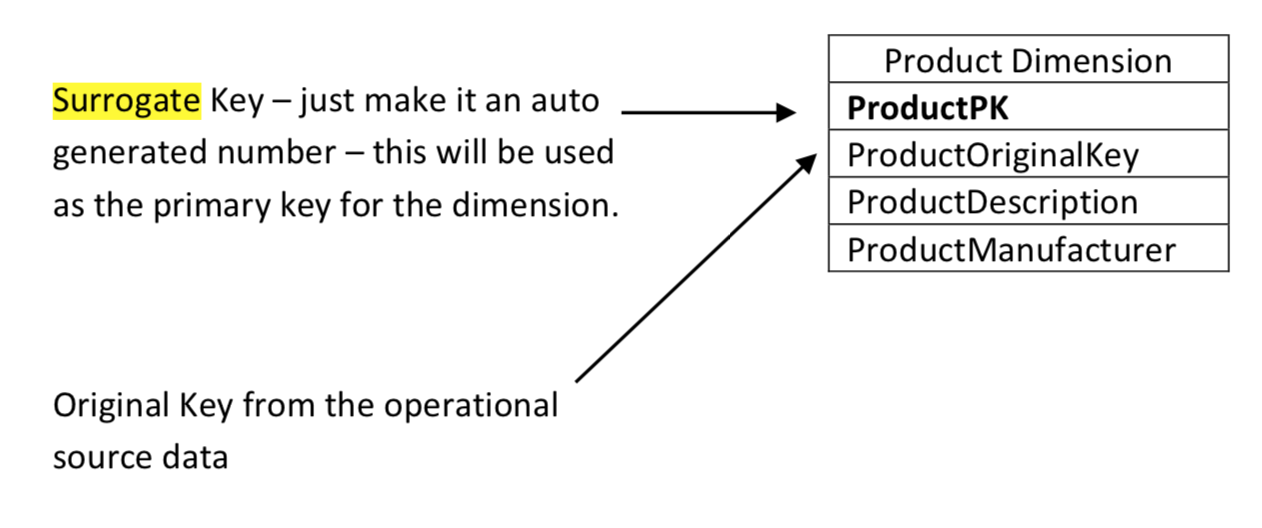
1. Separated into 2 categories
   1. 1. Measures
   2. 2. Dimensions
2. **Measures**
   1. Numeric data
   2. Can be aggregated depending how you select data
   3. Each Fact Table describes a specific set of discrete facts
      * But those facts are meaningless without context
      * So we need dimensions
3. **Dimension**
   1. Qualitative data, more descriptive than a Fact
4. Dimension Table = information about customer
   1. Like customer name, customer location

A screenshot of a cell phone

Description automatically generated

**Surrogate keys**

1. PK for a dimension table
2. Shouldn’t contain any information about contents of records
   1. Automatically increasing integers make good surrogate keys
3. Special keys are used for **date** and **time** dimensions
   1. But these keys differ from surrogate keys used for other dimension tables



**Factless Fact Table**

1. Fact table contains
   1. Key columns
      1. Key values that relate the measures to dimension tables(FK).
      2. All key columns combined make up the PK of fact table
   2. Measure columns
      1. Facts we want to analyze
      2. These are numbers
2. **Factless Fact Table** doesn’t contain measure columns unlike Fact Table

**Fully Additive Measures**

1. If a measure is additive across all dimensions
2. For example)
   1. Sales Quantity
   2. Sales dollor amount
   3. Sales dollor cost

**Semiadditive Measures**

1. Values that you can summarize across any related dimension except time.
2. For example)
   1. Stock levels. Changes daily. You can add up but it doesn’t make sense. You need to find the most recent value.

**Non-Additive Measures**

1. If unit is like
   1. %, ratios.
   2. numerator and denominator should be stored in a Fact Table
   3. Don’t store non-additive values in a fact table

**Degenerate Dimension(Fact Dimension)**

1. A screenshot of a cell phone

   Description automatically generated
2. In this Fact Table, there is a measure called ‘SalesOrderNumber’
   1. This is not a measure or metric or fact. Then why it’s in fact table?
   2. When program decide what to put into your dimensions or fact table, there might be some data that are neither facts nor dimensional attributes.
      * Reference numbers like…
        + Order numbers
        + Invoice numbers
        + Order line numbers
        + Etc…
   3. These attributes are useful in some types of analyses.
      * If you look for average number of products per order. You would have to relate the products to order number to calculate the average.
      * Attributes such as ‘SalesOrderNumber’ are called degenerate dimensions and these are kept as attributes of the fact table.

**Slowly Changing Dimensions (SCD)**

1. Data such as names(of existing product, customer) changes infrequently
2. 3 common methods to handle slowly changing dimensions
   1. Type 1: overwrite
      * Replaces old data with new value
      * Generally appropriate only when correcting error(misspell)
      * easy to maintain, but shows no historical data
   2. Type 2: Add a new dimension record
      * Makes data transformation more complex
      * Significantly increase the size of the dimension
   3. Type 3: Create new field
      * Is completely de-normalized.
      * Maintains history of values and effective date
      * Use it for small number of changes
        + E.g., **original value and current value without tracking interim changes**

**Handling Many to Many Relationships**

1. Method 1: Add a **Weighted Bridge Table**
2. Method 2: Lower the Grain of the Fact Table

**Aggregations and Storage Modes, Note 2**

**Cubes**

* A collection of one or more related measure groups and their associated dimensions
* To define a cube
  + Select a Fact Table from dimensional schema
  + Choose measures you want to work with
  + Select dimension tables that provide descriptions for the set of measures you selected

**Aggregations**

* Pre-calculated numeric data

**Physical Storage Options**

* OLAP DB use one of 3 methods to store multi-dimensional data
  + Relational OLAP (ROLAP)
  + Multidimensional OLAP (MOLAP)
  + Hybrid OLAP (HOLAP)
* Difference:
  + Way they store leaf-level data and aggregations
    - Leaf-level data
      * The finest grain of data defined in cube’s measure group.
      * So, leaf-level data corresponds to data of cube’s fact table
* ROLAP
  + Store detail value in relational **fact table**
  + Store aggregated values in relational **DB**
  + *Leave Original data in relational table and uses separate set of relational tables to store aggregates*
  + Advantage
    - Can handle large amounts of data
    - Data is not duplicated
  + Disadvantage
    - Performance = slow
    - Difficult to perform complex calculations
    - Need to create summary tables with materialized(indexed) views
* HOLAP
  + Leaves detail values in the relational **fact table**
  + But stores the aggregated values in **cube**
  + Data remains in relational tables, aggregations are stored on server in optimized multidimensional format
  + Advantage
    - HOLAP leverages cube tech for faster performance
    - When detail information is needed, HOLAP can “drill through” form cube into underlying relational data
    - Detail data is not duplicated
  + Disadvantage
    - Need to create summary tables with materialized(indexed) views
* MOLAP
  + Store both detail and aggregated value in **cube**
  + *Copies all data and aggregates to the analysis server in an optimized multidimensional format*
  + Advantage
    - Aggregates are stored in multi-dimensional form
      * Increasing performance
    - Optimized for query performance
  + Disadvantage
    - Limited in amount of data can handle
      * Because all calculations are performed when cube is built, it’s not possible to include a large amount of data in cube itself.

**Processing and Deploying cubes**

* Deploy:
  + “pushes” definition/structure of cube to the server
    - Roughly equivalent of issuing CREATE TABLE or ALTER TABLE SQL statements
    - Use this when you make changes to the structure of the cube
* Processing:
  + Populate the cube with data/aggregations from relational sources
    - Roughly equivalent of issuing INSERT or SELECT INTO SQL statements
    - Use this when you want to populate cube with fresh data
* The first time cube a deployed it must also be processed or it will contain no data

**Named Calculation, Calculated Members, Named sets**

**Named Calculation**

* In the data source view of SSAS, you can create **Named Calculation.**
* It’s a SQL expression represented as a calculated column in a table.
* It will appear and behave as a column in the table.
* Calculated members use MDX
* Calculated members store the definition in the cube
  + And values are calculated at query time
  + Therefore, changes or additions to calculated members require only redeployment and not a reprocess

**Named Set**

* It’s Multidimensional Expressions(MDX) that returns a set of dimension members.
* You can define **named sets** and save them as part of the cube definition.
* You create named sets by combining cube data, arithmetic operators, numbers and functions.

**KPI, Key Performance Indicators**

* It’s a metric that enable you to define a target value for a specific measure value.
* Example: Revenue
  + Every company has specific revenue targets
  + And they want to understand how they performs relative to that target
  + By defining KPI in cube, we can get a uniform representation of the indicator in various user interfaces
* For each KPI you develop expressions that calculate the KPI’s value, goal, current status, and trend. And define these elements of KPI
  + **VALUE** expression, Represent where the business is today.
    - Is a physical measure such as ‘Sales’, a calculated measure such as ‘Profit’
  + **GOAL** expression
    - Is a value, or MDX expression that resolves to a value that defines the target for the measure that value expression defines.
  + **STATUS** expression
    - Measures how VALUE compares to the GOAL
  + **TREND** expression, used to compare current STATUS value with value of STATUS at a previous point in time. (optional)

**A screenshot of a cell phone

Description automatically generated**

**ETL, Extract Transform Load process**

4 steps ETL Process

1. **Extraction**
   1. Bring raw data from operational data sources.
      1. Data source could be DB systems, spreadsheets, xml
   2. Usually we don’t do cleaning/transforming during this stage because we want the extraction to be as fast as possible and easy to restart if something goes wrong during process.
   3. Extracted data is stored into flat files or RDBMS
   4. This raw data can be kept permanently deleted after delivery stage or deleted after next extraction cycle
2. **Cleaning**
   1. We want to clean data
      1. because in most cases the raw data extracted from operational systems doesn’t not meet the strict requirements of the data warehouse. (dirty!)
      2. cleaning involves many discreet steps
         1. e.g., checking for valid postal codes, required fields, removing duplicates
      3. result = usually saved semi permanently as the transformations are difficult and irreversible
3. **Conforming**
   1. Is required if 2 or more operational data sources are merged
   2. Separate data sources can’t be queried together unless some or all textual labels in these sources have been made identical and unless similar numeric measures
4. **Delivery**
   1. Load the cleaned/conformed data into data warehouse DB

**Implementing Security**

**Roles**

* To control access to various objects such as DB, Dimensions, Cubes.
* **Server role**
  + Broadest admin right are granted
* **DB roles**
  + Limited privileges
* By granting DB role full control
  + Right to CREATE, MODIFY, DELETE, PROCESS any object in DB, EXECUTE backups and DB level trace, MANAGE DB security, access all data in DB
* Whitelist
  + allowed set, member role is permiited to access
* blacklist
  + denied set
* Restricting access to cells
  + By default, all cells within a cube to members of DB role have been granted access are readable.
  + But you can define access on a cell-by-cell
  + When enabled, level of access granted to a cell is evaluated using logical MDX expression.
  + If you want to limit data access to only data from a specific fiscal year…
    - (NOT [Measures].CurrentMember IS [Measures].[Amount]) OR ([Date].[Fiscal Year].CurrentMember IS [Date].[Fiscal Year].[FY 2009])

**Data Mining Concepts**

Data mining is frequently descrbied as “process of extracting valid, authentic and actionable information from large DB”. Data mining derives patterns and trends that exist in data. Mining models can be applied in specific business scenarios.

* Forecasting sales
* Targeting mailings toward specific customers
* Determining which products are likely to be sold together

**6 Basic Steps**

1. **Defining the problem**
   1. Clearly define business problem
      1. Analyzing business requirements
      2. Defining scope of the problem
      3. Defining metrics by which model will be evaluated
      4. Defining final objective for data mining project
   2. These tasks translate into questions such as…
      1. What are you looking for?
      2. Which attribute of the dataset do you want to try to predict?
      3. What types of relationships are you trying to find?
      4. Do you want to make predictions from data mining model or just look for interesting patterns and associations?
      5. How is data distributed?
      6. How are the columns related, or if there are multiple tables, how are tables related?
   3. If data does not support the needs of the users, you may have to redefine the project
2. **Preparing Data**
   1. 2nd step is to consolidate and clean the data that was identified in defining the problem step
   2. data can be scattered across a company and stored in different formats. it might have some errors.
   3. You have to use some form of automation, to explore data and fine inconsistencies.
3. **Exploring Data**
   1. 3rd step is to explore prepared data
   2. You must understand data in order to make appropriate decisions when you create the models. Exploration techniques include calculating minimum and maximum values.
   3. After exploring data
      1. You can decide if the dataset contains flawed data, and then you can devise a strategy for fixing problems
   4. Building Models
      1. 4th step in data mining process is to build mining models
      2. Before you build a model,
         1. You must randomly separate prepared data into separate training and testing datasets.
         2. Use training dataset to build the model
         3. And testing dataset to test the accuracy of model by creating prediction queries.
         4. You will use knowledge that you gain from **Exploring Data** step to help define and create a mining model. **A model typically contains input columns, an identifying column, and predictable column.**
      3. After define structure of mining model
         1. Process it
         2. Populating empty structure with the patterns that describe the model.
            1. This is known as **training** the model
         3. Patterns are found by passing original data through a mathematical algorithm.
         4. A mining model is defined by a **data mining structure object,** **data mining model object**, and **data mining algorithm.**
4. **Exploring and Validating models**
   1. 5th step in data mining process is to explore the models you built and test their effectiveness
   2. Don’t deploy a model into a production environment without 5th testing how well the model performs. You may have created several models and will have to decide which model will perform the best. If none of the models that you created in the **Building Models** step perform well, you may have to return to a previous step in the process, either by redefining problem or by reinvestigating the data in the original dataset.
5. **Deploying and Updating Models**
   1. Last step in data mining process is to deploy to production environment
   2. Updating model is part of the deployment strategy. As more data comes into organization, you must reprocess the models, thereby improving their effectiveness.

**Decision Trees**

* Useful for predicting exact outcomes. Applying the decision trees algorithm to a training results in the formation of a tree that allows the user to map a path to a successful outcome.
* Decision trees algorithm would be useful for a bank that wants to ascertain the characteristics of good customers. In this case, predict outcome is whether or not the applicant represents a bad credit risk. Outcome of a decision tree may be a yes/no result or list of numeric values, with each value assigned a probability

**Clustering**

* Is different from decision trees in that it involves grouping data into meaningful clusters with no specific outcome. It goes through a looped process whereby it reevaluates each cluster against all the other clusters looking for patterns in data.
* The algorithm is useful when a large DB with undreds of attributes is first evaluated.
* The process may uncover a relationship between data items that was never suspected. In the case of the bank that wants to determine credit risk, clustering might be used to identify groups of similar customers. It could reveal that certain customer attributes are more meaningful than originally thought.

**Midterm questions**

When data has changed in data warehouse, you must?

* Process the cube

Which sequence of jobs would you follow to load data in to the data warehouse?

* First load data into dimension tables, then fact tables, and Aggregates if any

True statements

* Dimension tables contain only a single column PK
* Dimensions should contain a surrogate key
* Dimensions contain a single index based on PK
* In a Star Schema dimension tables do contain FK

What is a hybrid schema and why would you use one?

* A schema where you normalize only part of the design to accommodate shared dimensions

A screenshot of a cell phone

Description automatically generated