

DATA WAREHOUSE NOTES



September 25, 2016

Table of Contents

[1. Database 3](#_Toc462570837)

[1.1 Entity and Attribute 3](#_Toc462570838)

[1.1.1 Types of Attributes 3](#_Toc462570839)

[1.2 Keys 4](#_Toc462570840)

[1.2.1 Types of Keys 4](#_Toc462570841)

[1.3 Types of Relationships 7](#_Toc462570842)

[1.3.1 One-to-Many Relationship 7](#_Toc462570843)

[1.3.2 Many-to-Many Relationships 7](#_Toc462570844)

[1.3.3 One-to-One Relationships 8](#_Toc462570845)

[2. Data Modelling 9](#_Toc462570846)

[2.1. Levels of Data Model 9](#_Toc462570847)

[2.1.1. Conceptual Data Model 9](#_Toc462570848)

[2.1.2. Logical Data Model 9](#_Toc462570849)

[2.1.3. Physical Data Model 10](#_Toc462570850)

[2.2. Components of Data Model 10](#_Toc462570851)

[2.3. Types of Data Model 10](#_Toc462570852)

[2.3.1. Object Based Data Models 11](#_Toc462570853)

[2.3.2. Physical Data Models 11](#_Toc462570854)

[2.3.3. Record Based Logical Models 11](#_Toc462570855)

[2.4. Dimensional Data Model 11](#_Toc462570856)

[Star Schema 12](#_Toc462570857)

[Snowflake Schema 13](#_Toc462570858)

[3. Data Warehouse 13](#_Toc462570859)

[3.1. Operational Data Store (ODS) 14](#_Toc462570860)

[3.2. Data Warehouse Architecture Layers 14](#_Toc462570861)

[Data Source Layer 15](#_Toc462570862)

[Data Extraction Layer 15](#_Toc462570863)

[Staging Area 15](#_Toc462570864)

[ETL Layer 15](#_Toc462570865)

[Data Storage Layer 15](#_Toc462570866)

[Data Logic Layer 15](#_Toc462570867)

[Data Presentation Layer 15](#_Toc462570868)

[Metadata Layer 16](#_Toc462570869)

[System Operations Layer 16](#_Toc462570870)

[3.3. Data warehouse / Data Mart / ODS 16](#_Toc462570871)

[4. ETL 17](#_Toc462570872)

[4.1. OLAP v/s OLTP 17](#_Toc462570873)

[5. Data Mapping 17](#_Toc462570874)

[6. Reference 17](#_Toc462570875)

# Database

*Watch first two videos of Dr Daniel Soper on Database.*

## Entity and Attribute

Entities are the things or objects in our system. They represent conceptual categories, not specific objects.

For example - In a store there may be a shop assistant called Eve. When she quits and another person called James is hired, he has the same role as Eve: he is a shop assistant. Moreover, the store can also hire other shop assistants. In our system, we will only have the general entity *Shop Assistant*. This entity is the role that both Eve and James have. So, Eve and James are specific instances of the same general concept of a “Shop Assistant”. *Shop Assistant* is the entity in our system.

Each entity is associated with some details known as attributes. attributes are additional information about a specific instance of an entity.

For Example - In the previous case, the entity is Shop Assistant and can have attributes like first and last names, a date of birth, eye color, and so on.

### Types of Attributes

#### Single valued Attributes

An attribute, that has a single value for a particular entity is known as single valued attributes. For example, age of an employee entity.

#### Multi valued Attributes

An attributes that may have multiple values for the same entity is known as multi valued attributes. For example, colors of a car entity.

#### Simple Attributes/Atomic Attributes

The attributes which cannot be divided into smaller subparts are called simple or atomic attributes. For example, age of employee entity.

#### Compound Attribute/Composite Attribute

Attribute can be subdivided into two or more other Attribute. For Example, Name can be divided into First name, Middle name and Last name.

#### Stored Attribute

An attribute, which cannot be derived from other attribute, is known as stored attribute. For example, BirthDate of employee.

#### Derived Attribute

Attributes derived from other stored attribute. For example, age from Date of Birth and Today’s date.

#### Complex Attributes

If an attribute for an entity, is built using composite and multivalued attributes, then these attributes are called complex attributes. For example, a person can have more than one residence and each residence can have multiple phones, an addressphone for a person entity can be specified as –

*{Addressphone (phone {(Area Code, Phone Number)}, Address(Sector Address (Sector Number,House Number),*

*City, State, Pin))}*

Here {} are used to enclose multivalued attributes and () are used to enclose composite attributes with comma separating individual attributes.

#### Key Attribute

It represents primary key. (main characteristics of an entity). It is an attribute, that has distinct value for each entity/element in an entity set. For example, Roll number in a Student Entity Type.

#### Non Key Attributes

These are attributes other than candidate key attributes in a table. For example, Firstname is a non key attribute as it does not represent the main characteristics of the entity.

#### Required Attribute

A required attribute is an attribute that must have a data value. These attributes are required because they describe what is important in the entity. For example, in a STUDENT entity, firstname and lastname is a required attribute.

#### Optional Attribute/Null Value Attribute

An optional attribute may not have a value in it and can be left blank. For example, in a STUDENT entity, Middlename or email address is an optional attribute. as some students may not have middlename or email address.

## Keys

There are three main types of keys, candidate keys, primary keys and foreign keys. There is also an alternative key or secondary key that can be used, as the name suggests, as a secondary or alternative key to the primary key.

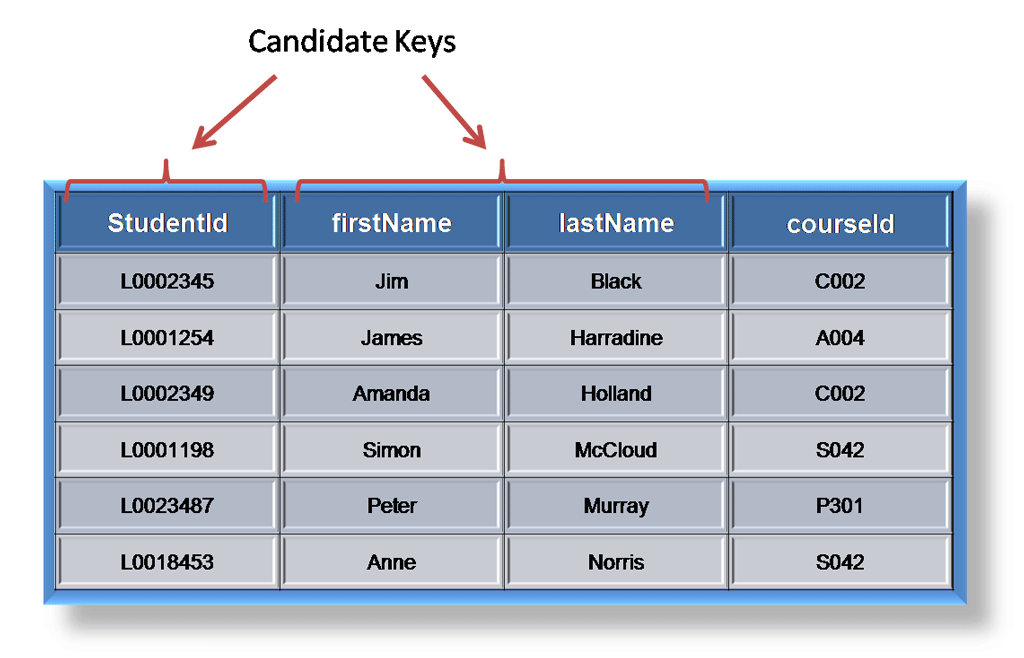
### Types of Keys

#### Super Key

A Super key is any combination of fields within a table that uniquely identifies each record within that table.

#### Candidate Key

A candidate is a subset of a super key. A candidate key is a single field or the least combination of fields that uniquely identifies each record in the table. The least combination of fields distinguishes a candidate key from a super key. Every table must have at least one candidate key but at the same time can have several.



As an example we might have a student\_id that uniquely identifies the students in a student table. This would be a candidate key. But in the same table we might have the student’s first name and last name that also, when combined, uniquely identify the student in a student table. These would both be candidate keys.

In order to be eligible for a candidate key it must pass certain criteria.

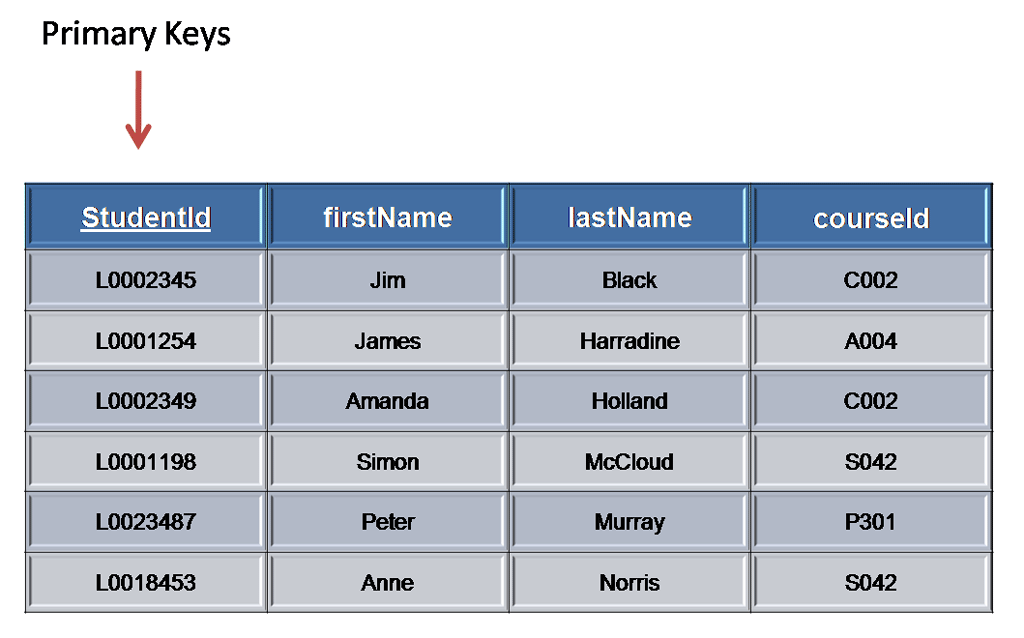
* It must contain unique values
* It must not contain null values
* It contains the minimum number of fields to ensure uniqueness
* It must uniquely identify each record in the table

Once your candidate keys have been identified you can now select one to be your primary key.

#### Primary Key

A primary key is a candidate key that is most appropriate to be the main reference key for the table. As its name suggests, it is the primary key of reference for the table and is used throughout the database to help establish relationships with other tables. As with any candidate key the primary key must contain unique values, must never be null and uniquely identify each record in the table.

As an example, a student id might be a primary key in a student table, a department code in a table of all departments in an organization. This module has the code DH3D 35 that is no doubt used in a database somewhere to identify RDBMS as a unit in a table of modules. In the table below we have selected the candidate key student\_id to be our most appropriate primary key.

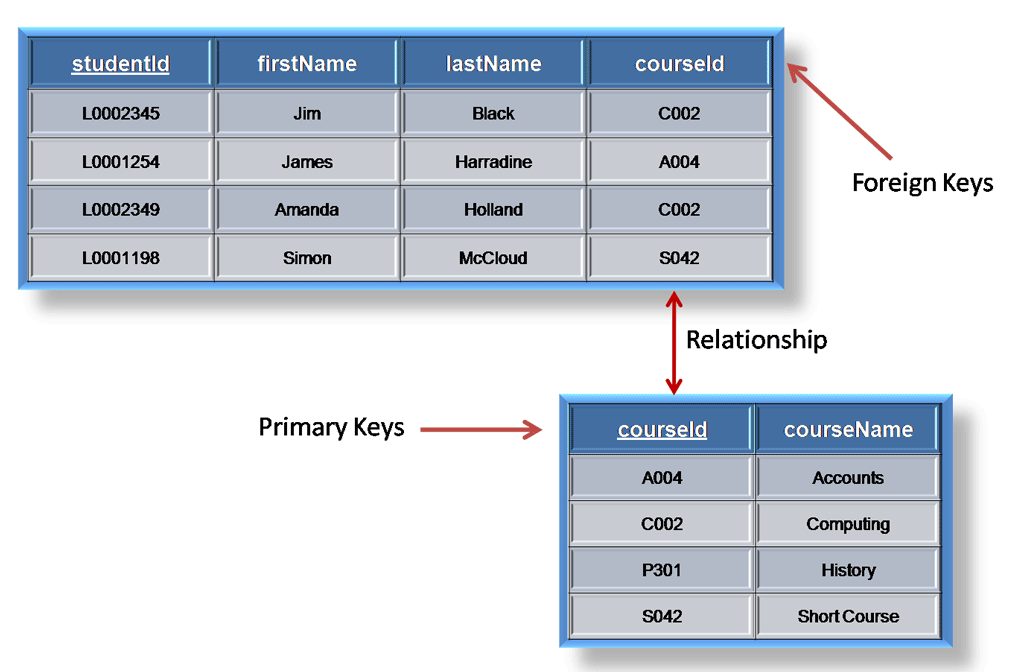


Primary keys are mandatory for every table each record must have a value for its primary key. When choosing a primary key from the pool of candidate keys always choose a single simple key over a composite key.

#### Foreign Key

A foreign key is generally a primary key from one table that appears as a field in another where the first table has a relationship to the second. In other words, if we had a table A with a primary key X that linked to a table B where X was a field in B, then X would be a foreign key in B.

An example might be a student table that contains the course\_id the student is attending. Another table lists the courses on offer with course\_id being the primary key. The 2 tables are linked through course\_id and as such course\_id would be a foreign key in the student table.



#### Secondary Key or Alternative Key

A table may have one or more choices for the primary key. Collectively these are known as candidate keys as discuss earlier. One is selected as the primary key. Those not selected are known as secondary keys or alternative keys.

For example, in the table showing candidate keys above we identified two candidate keys, studentId and firstName + lastName. The studentId would be the most appropriate for a primary key leaving the other candidate key as secondary or alternative key. It should be noted for the other key to be candidate keys, we are assuming you will never have a person with the same first and last name combination. As this is unlikely we might consider fistName+lastName to be a suspect candidate key as it would be restrictive of the data you might enter. It would seem a shame to not allow John Smith onto a course just because there was already another John Smith.

#### Simple Key

Any of the keys described before (i.e. primary, secondary or foreign) may comprise one or more fields, for example if firstName and lastName was our key this would be a key of two fields where as studentId is only one. A simple key consists of a single field to uniquely identify a record. In addition, the field in itself cannot be broken down into other fields, for example, studentId, which uniquely identifies a particular student, is a single field and therefore is a simple key. No two students would have the same student number.

#### Compound Key

A compound key consists of more than one field to uniquely identify a record. A compound key is distinguished from a composite key because each field, which makes up the primary key, is also a simple key in its own right. An example might be a table that represents the modules a student is attending. This table has a studentId and a moduleCode as its primary key. Each of the fields that make up the primary key are simple keys because each represents a unique reference when identifying a student in one instance and a module in the other.

#### Composite

A composite key consists of more than one field to uniquely identify a record. This differs from a compound key in that one or more of the attributes, which make up the key, are not simple keys in their own right. Taking the example from compound key, imagine we identified a student by their firstName + lastName. In our table representing students on modules our primary key would now be firstName + lastName + moduleCode. Because firstName + lastName represent a unique reference to a student, they are not each simple keys, they have to be combined in order to uniquely identify the student. Therefore, the key for this table is a composite key.

## Types of Relationships

### One-to-Many Relationship

A one-to-many relationship is the most common type of relationship. In this type of relationship, a row in table A can have many matching rows in table B, but a row in table B can have only one matching row in table A. For example, the publishers and titles tables have a one-to-many relationship: each publisher produces many titles, but each title comes from only one publisher.

Make a one-to-many relationship if only one of the related columns is a primary key or has a unique constraint.

The primary key side of a one-to-many relationship is denoted by a key symbol. The foreign key side of a relationship is denoted by an infinity symbol.

### Many-to-Many Relationships

In a many-to-many relationship, a row in table A can have many matching rows in table B, and vice versa. You create such a relationship by defining a third table, called a junction table, whose primary key consists of the foreign keys from both table A and table B. For example, the authors table and the titles table have a many-to-many relationship that is defined by a one-to-many relationship from each of these tables to the titleauthors table. The primary key of the titleauthors table is the combination of the au\_id column (the authors table's primary key) and the title\_id column (the titles table's primary key).

### One-to-One Relationships

In a one-to-one relationship, a row in table A can have no more than one matching row in table B, and vice versa. A one-to-one relationship is created if both of the related columns are primary keys or have unique constraints.

This type of relationship is not common because most information related in this way would be all in one table. You might use a one-to-one relationship to:

* Divide a table with many columns.
* Isolate part of a table for security reasons.
* Store data that is short-lived and could be easily deleted by simply deleting the table.
* Store information that applies only to a subset of the main table.

The primary key side of a one-to-one relationship is denoted by a key symbol. The foreign key side is also denoted by a key symbol.

Data Modelling

Data Modelling is the analysis of data objects and their relationships to other data objects. It is often the first step in database design and object-oriented programming. It is a process used to define and analyze data requirements needed to support the business processes within the scope of corresponding information systems in organizations.

This is a very important step in the data warehousing project. Indeed, it is fair to say that the foundation of the data warehousing system is the data model. A good data model will allow the data warehousing system to grow easily, as well as allowing for good performance.

## Levels of Data Model

The three levels/types of data modeling are -

### Conceptual Data Model

A conceptual data model identifies the highest-level relationships between the different entities. Features of conceptual data model include –

* Includes the important entities and the relationships among them.
* No attribute is specified.
* No primary key is specified.

### Logical Data Model

A logical data model describes the data in as much detail as possible, without regard to how they will be physical implemented in the database.

Features of a logical data model include –

* Includes all entities and relationships among them.
* All attributes for each entity are specified.
* The primary key for each entity is specified.
* Foreign keys (keys identifying the relationship between different entities) are specified.
* Normalization occurs at this level.

The steps for designing the logical data model are as follows –

* Specify primary keys for all entities.
* Find the relationships between different entities.
* Find all attributes for each entity.
* Resolve many-to-many relationships.
* Normalization.

Comparison with the Conceptual Data Model diagram –

* In a logical data model, primary keys are present, whereas in a conceptual data model, no primary key is present.
* In a logical data model, all attributes are specified within an entity. No attributes are specified in a conceptual data model.
* Relationships between entities are specified using primary keys and foreign keys in a logical data model. In a conceptual data model, the relationships are simply stated, not specified, so we simply know that two entities are related, but we do not specify what attributes are used for this relationship.

### Physical Data Model

Physical data model represents how the model will be built in the database. A physical database model shows all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables.

Features of a physical data model include –

* Specification all tables and columns.
* Foreign keys are used to identify relationships between tables.
* De-normalization may occur based on user requirements.
* Physical considerations may cause the physical data model to be quite different from the logical data model.
* Physical data model will be different for different RDBMS. For example, data type for a column may be different between MySQL and SQL Server.

The steps for physical data model design are as follows –

* Convert entities into tables.
* Convert relationships into foreign keys.
* Convert attributes into columns.
* Modify the physical data model based on physical constraints / requirements.

Comparing with the logical data model diagram –

* Entity names are now table names.
* Attributes are now column names.
* Data type for each column is specified. Data types can be different depending on the actual database being used.

## Components of Data Model

A data model comprises of three components –

* A structural part, consisting of a set of rules according to which databases can be constructed.
* A manipulative part, defining the types of operation that are allowed on the data (this includes the operations that are used for updating or retrieving data from the database and for changing the structure of the database).
* Possibly a set of integrity rules, which ensures that the data is accurate.

## Types of Data Model

They fall into three broad categories –

* Object Based Data Models
* Physical Data Models
* Record Based Data Models

The object based and record based data models are used to describe data at the conceptual and external levels, the physical data model is used to· describe data at the internal level.

### Object Based Data Models

Object based data models use concepts such as entities, attributes, and relationships. An entity is a distinct object (a person, place, concept, and event) in the organization that is to be represented in the database. An attribute is a property that describes some aspect of the object that we wish to record, and a relationship is an association between entities.

Some of the more common types of object based data model are –

* Entity-Relationship
* Object Oriented
* Semantic
* Functional

The Entity-Relationship model has emerged as one of the main techniques for modeling database design and forms the basis for the database design methodology. The object oriented data model extends the definition of an entity to include, not only the attributes that describe the state of the object but also the actions that are associated with the object, that is, its behavior. The object is said to encapsulate both state and behavior. Entities in semantic systems represent the equivalent of a record in a relational system or an object in an OO system but they do not include behaviour (methods). They are abstractions 'used to represent real world (e.g. customer) or conceptual (e.g. bank account) objects. The functional data model is now almost twenty years old. The original idea was to' view the database as a collection of extensionally defined functions and to use a functional language for querying the database.

### Physical Data Models

Physical data models describe how data is stored in the computer, representing information such as record structures, record ordering, and access paths. There are not as many physical data models as logical data models, the most common one being the Unifying Model.

### Record Based Logical Models

Record based logical models are used in describing data at the logical and view levels. In contrast to object based data models, they are used to specify the overall logical structure of the database and to provide a higher-level description of the implementation. Record based models are so named because the database is structured in fixed format records of several types. Each record type defines a fixed number of fields, or attributes, and each field is usually of a fixed length.

The three most widely accepted record based data models are –

* Hierarchical Model
* Network Model
* Relational Model

## Dimensional Data Model

Dimensional data model is most often used in data warehousing systems. This is different from the 3rd normal form, commonly used for transactional (OLTP) type systems. As you can imagine, the same data would then be stored differently in a dimensional model than in a 3rd normal form model.

To understand dimensional data modeling, let's define some of the terms commonly used in this type of modeling –

* Dimension: A category of information. For example, the time dimension.
* Attribute: A unique level within a dimension. For example, Month is an attribute in the Time Dimension.
* Hierarchy: The specification of levels that represents relationship between different attributes within a dimension. For example, one possible hierarchy in the Time dimension is Year → Quarter → Month → Day.

**Fact Table** - A fact table is a table that contains the measures of interest. For example, sales amount would be such a measure. This measure is stored in the fact table with the appropriate granularity. For example, it can be sales amount by store by day. In this case, the fact table would contain three columns: A date column, a store column, and a sales amount column.

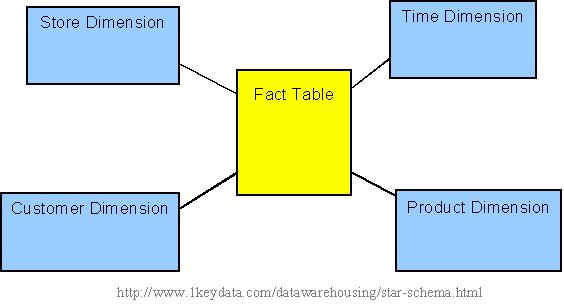
**Lookup Table** - The lookup table provides the detailed information about the attributes. For example, the lookup table for the Quarter attribute would include a list of all of the quarters available in the data warehouse. Each row (each quarter) may have several fields, one for the unique ID that identifies the quarter, and one or more additional fields that specifies how that particular quarter is represented on a report (for example, first quarter of 2001 may be represented as "Q1 2001" or "2001 Q1").

A dimensional model includes fact tables and lookup tables. Fact tables connect to one or more lookup tables, but fact tables do not have direct relationships to one another. Dimensions and hierarchies are represented by lookup tables. Attributes are the non-key columns in the lookup tables.

In designing data models for data warehouses / data marts, the most commonly used schema types are –

### Star Schema

In the star schema design, a single object (the fact table) sits in the middle and is radically connected to other surrounding objects (dimension lookup tables) like a star. Each dimension is represented as a single table. The primary key in each dimension table is related to a foreign key in the fact table.

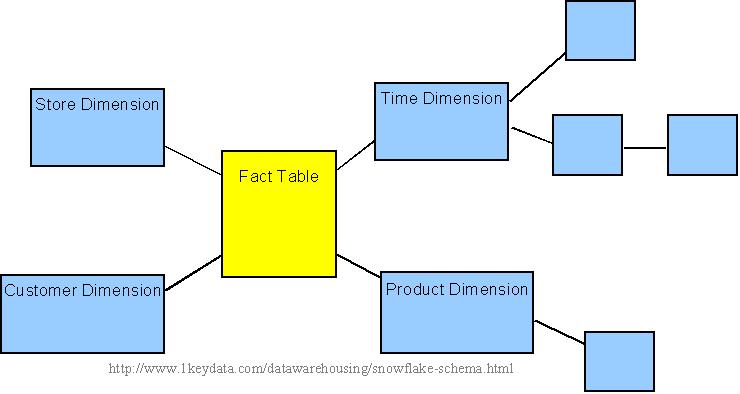


All measures in the fact table are related to all the dimensions that fact table is related to. In other words, they all have the same level of granularity. A star schema can be simple or complex. A simple star consists of one fact table; a complex star can have more than one fact table.

Let's look at an example: Assume our data warehouse keeps store sales data, and the different dimensions are time, store, product, and customer. In this case, the figure on the left represents our star schema. The lines between two tables indicate that there is a primary key / foreign key relationship between the two tables. Note that different dimensions are not related to one another.

### Snowflake Schema

he snowflake schema is an extension of the star schema, where each point of the star explodes into more points. In a star schema, each dimension is represented by a single dimensional table, whereas in a snowflake schema, that dimensional table is normalized into multiple lookup tables, each representing a level in the dimensional hierarchy.



For example, the Time Dimension that consists of 2 different hierarchies:

1. Year → Month → Day

2. Week → Day

We will have 4 lookup tables in a snowflake schema: A lookup table for year, a lookup table for month, a lookup table for week, and a lookup table for day. Year is connected to Month, which is then connected to Day. Week is only connected to Day. A sample snowflake schema illustrating the above relationships in the Time Dimension is shown to the right.

The main advantage of the snowflake schema is the improvement in query performance due to minimized disk storage requirements and joining smaller lookup tables. The main disadvantage of the snowflake schema is the additional maintenance efforts needed due to the increase number of lookup tables.

# Data Warehouse

A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision

on making process.

* Subject-Oriented: A data warehouse can be used to analyze a particular subject area. For example, "sales" can be a particular subject.
* Integrated: A data warehouse integrates data from multiple data sources. For example, source A and source B may have different ways of identifying a product, but in a data warehouse, there will be only a single way of identifying a product.
* Time-Variant: Historical data is kept in a data warehouse. For example, one can retrieve data from 3 months, 6 months, 12 months, or even older data from a data warehouse. This contrasts with a transactions system\*, where often only the most recent data is kept. For example, a transaction system may hold the most recent address of a customer, where a data warehouse can hold all addresses associated with a customer.
* Non-volatile: Once data is in the data warehouse, it will not change. So, historical data in a data warehouse should never be altered.

## Types of Data Architecture

1. Centralized Architecture
2. Independent Data Mart Architecture
3. Data Mart Bus architecture
4. Hub and Spoke
5. Federated Architecture

Data warehouse can be designed in two ways -

**Bottom-up design** - In the bottom-up approach, data marts are first created to provide reporting and analytical capabilities for specific business processes. These data marts can then be integrated to create a comprehensive data warehouse. It involves low initial cost, takes less time and is difficult to maintain.

**Top-down design** - The top-down approach is designed using a normalized enterprise data model. "Atomic" data, that is, data at the greatest level of detail, are stored in the data warehouse. Dimensional data marts containing data needed for specific business processes or specific departments are created from the data warehouse. It involves high initial cost and is time Consuming but easy to maintain.

## Operational Data Store (ODS)

An operational data store (ODS) is a type of database that's often used as an interim logical area for a data warehouse.

* An ODS is targeted for the lowest granular queries whereas a data warehouse is usually used for complex queries against summary-level or on aggregated data
* An ODS is meant for operational reporting and supports current or near real-time reporting requirements whereas a data warehouse is meant for historical and trend analysis reporting usually on a large volume of data
* An ODS contains only a short window of data, while a data warehouse contains the entire history of data
* An ODS provides information for operational and tactical decisions on current or near real-time data while a data warehouse delivers feedback for strategic decisions leading to overall system improvements
* In an ODS the frequency of data load could be every few minutes or hourly whereas in a data warehouse the frequency of data loads could be daily, weekly, monthly or quarterly

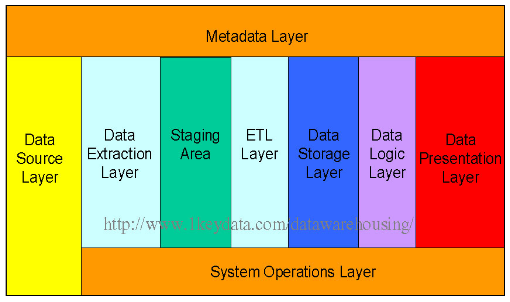
## Data Warehouse Architecture Layers

Different data warehousing systems have different structures. Some may have an ODS (operational data store), while some may have multiple data marts. Some may have a small number of data sources, while some may have dozens of data sources. In view of this, it is far more reasonable to present the different layers of a data warehouse architecture rather than discussing the specifics of any one system.

In general, all data warehouse systems have the following layers -

* Data Source Layer
* Data Extraction Layer
* Staging Area
* ETL Layer
* Data Storage Layer
* Data Logic Layer
* Data Presentation Layer
* Metadata Layer
* System Operations Layer

The picture below shows the relationships among the different components of the data warehouse architecture



### Data Source Layer

This represents the different data sources that feed data into the data warehouse. The data source can be of any format -- plain text file, relational database, other types of database, Excel file, etc., can all act as a data source. Many different types of data can be a data source:

* Operations -- such as sales data, HR data, product data, inventory data, marketing data, systems data.
* Web server logs with user browsing data.
* Internal market research data.
* Third-party data, such as census data, demographics data, or survey data.

All these data sources together form the Data Source Layer.

### Data Extraction Layer

Data gets pulled from the data source into the data warehouse system. There is likely some minimal data cleansing, but there is unlikely any major data transformation.

### Staging Area

This is where data sits prior to being scrubbed and transformed into a data warehouse / data mart. Having one common area makes it easier for subsequent data processing / integration.

### ETL Layer

This is where data gains its "intelligence", as logic is applied to transform the data from a transactional nature to an analytical nature. This layer is also where data cleansing happens. The ETL design phase is often the most time-consuming phase in a data warehousing project, and an ETL tool is often used in this layer.

### Data Storage Layer

This is where the transformed and cleansed data sit. Based on scope and functionality, 3 types of entities can be found here: data warehouse, data mart, and operational data store (ODS). In any given system, you may have just one of the three, two of the three, or all three types.

### Data Logic Layer

This is where business rules are stored. Business rules stored here do not affect the underlying data transformation rules, but do affect what the report looks like.

### Data Presentation Layer

This refers to the information that reaches the users. This can be in a form of a tabular / graphical report in a browser, an emailed report that gets automatically generated and sent everyday, or an alert that warns users of exceptions, among others. Usually an OLAP tool and/or a reporting tool is used in this layer.

### Metadata Layer

This is where information about the data stored in the data warehouse system is stored. A logical data model would be an example of something that's in the metadata layer. A metadata tool is often used to manage metadata.

### System Operations Layer

This layer includes information on how the data warehouse system operates, such as ETL job status, system performance, and user access history.

## Data warehouse / Data Mart / ODS

A data warehouse is where you store data from multiple data sources to be used for historical and trend analysis reporting. It acts as a central repository for many subject areas and contains the “single version of truth”. A data mart serves the same purpose but comprises only one subject area.

The purpose of an ODS is to integrate corporate data from different heterogeneous data sources in order to facilitate operational reporting in real-time or near real-time. Usually data in the ODS will be structured similar to the source systems, although during integration the data can be cleaned, de-normalized, and business rules applied to ensure data integrity. This integration will happen at the lowest granular level and occur quite frequently throughout the day. Normally an ODS will not be optimized for historical and trend analysis as this is left to the data warehouse.

## Data Curation

Data curation is the management of data throughout its lifecycle, from creation and initial storage to the time when it is archived for posterity or becomes obsolete and is deleted. The main purpose of data curation is to ensure that data is reliably retrievable for future research purposes or reuse. Within the enterprise, compliance is another primary purpose.

## Data Profiling

Data profiling is a process of examining available data in the existing data source and collect statistics and information about the data, improper data profiling can lead to poor quality data and this could adversely affect the overall purpose of the information system.

There are multiple things one does as a part of Data Profiling at the attribute level.

1. Completeness Analysis - How often is a given attribute populated, versus blank or null?
2. Uniqueness Analysis - How many unique (distinct) values are found for a given attribute across all records? Are there duplicates? Should there be?
3. Values Distribution Analysis - What is the distribution of records across different values for a given attribute?
4. Range Analysis - What are the minimum, maximum, average and median values found for a given attribute?
5. Pattern Analysis - What formats were found for a given attribute, and what is the distribution of records across these formats?

The purpose of this analysis and statistics is to

* Find out whether existing data can easily be used for other purposes
* Analyze the customer’s characteristics, value and pattern in their changing behaviors.
* Improve the ability to search the data by tagging it with keywords, descriptions, or assigning it to a category
* Give metrics on data quality including whether the data conforms to particular standards or patterns
* Assess the risk involved in integrating data for new applications, including the challenges of joins
* Discover metadata of the source database, including value patterns and distributions, key candidates, foreign-key candidates, and functional dependencies.
* Assess whether known metadata accurately describes the actual values in the source database
* Understanding data challenges early in any data intensive project, so that late project surprises are avoided. Finding data problems late in the project can lead to delays and cost overruns.
* Have an enterprise view of all data, for uses such as master data management where key data is needed, or data governance for improving data quality

Data profiling can add value in a wide variety of situations. The basic litmus test is, “Is the quality of data important for this initiative?” If the answer is “yes”, then data profiling can help as it can quickly and thoroughly unveil the true content and structure of your data.

Some example scenarios include: -

1. Data Warehousing / Business Intelligence (DW/BI) projects

These projects involve gathering data from disparate systems for the purpose of reporting and analysis. Data profiling can help ensure project success by –

* Identifying data quality issues that must be corrected in the source system
* Identifying issues that can be corrected in ETL processing
* Discovering unanticipated business rules
* Even potentially providing a “no-go” decision on the project as a whole
* Data conversion / Migration projects

These involve moving data from a legacy system to a new system. Data profiling can help reduce project risk by:

* Identifying data quality issues that must be handled in the code that moves data from the legacy system to the new system
* Identifying data issues that may require a change to the target (new) system
* Source System Data Quality Initiative

These projects endeavor to assess and improve the data quality of a given source system, seeking to fix existing issues as well as avoid those issues in the future. Data profiling can help maximize project ROI by

* Highlighting the areas within the system suffering from the most serious and/or numerous data quality issues
* Identifying issues that may be the result of bad user input or errant system interfaces

Tools used are –

* Informatica Data quality
* Datiris Profiler

ETL

Extract, Load, Transform (ELT) is a process where data is extracted for the source, then loaded into a staging table in the database, transforming it where it sits in the database and then loading it into the target database or data warehouse.

## OLAP v/s OLTP

<http://datawarehouse4u.info/OLTP-vs-OLAP.html>

# Data Mapping

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.8056&rep=rep1&type=pdf>

# Reference

<http://www.1keydata.com/datawarehousing/what-is-olap.html>

<https://web.archive.org/web/20100106080548/http://www.olapreport.com/FASMI.HTM>

<https://www.tutorialspoint.com/dwh/dwh_interview_questions.htm>

<https://dwbi.org/data-modelling/dimensional-model/58-top-50-dwbi-interview-questions-with-answers>