**CSC 303**

**DISTRIBUTED DATABASE SYSTEM**

* A **distributed database** (DDB) is a collection of multiple, logically interrelated databasesdistributed over a computer network

• A **distributed database management system** (DDBMS) is the software that managesthe DDB and provides an access mechanism that makes this distribution transparent tothe users. Note however that the terms are most times, used interchangeably

Distributed database design defines the optimum allocation strategy for database fragments in order to ensure database integrity, security, and performance. The allocation strategy determines how to partition the database and where to store each fragment.

A database may need to be distributed among multiplegeographically disperse locations. Processes that access the database may also vary from one location to another. Resources that can be distributed includes but not limited to:

* Functions
* Processes
* Data
* Control

Forexample, a retail process and a warehouse storage process are likely to be found in different physical locations. If thedatabase data and processes are to be distributed across the system, portions of a database, known as databasefragments, may reside in several physical locations. A **database fragment** is a subset of a database that is stored ata given location. The database fragment may be composed of a subset of rows or columns from one or multiple tables.

**DATABASE SOFTWARE SELECTION**

The selection of DBMS software is critical to the information system’s smooth operation.To avoid false expectations,

the end user must be made aware of the limitations of both the DBMS and the database. The followings are factors that should be considered while purchasing a DBMS and Database Application:

* *Cost*. This includes the original purchase price, along with maintenance, operational, license, installation,

training, and conversion costs.

* *DBMS features and tools*. Some database software includes a variety of tools that facilitate the application

development task. For example, the availability of query by example (QBE), screen painters, report generators,

application generators, data dictionaries, and so on, helps to create a more pleasant work environment for

both the end user and the application programmer. Database administrator facilities, query facilities, ease of

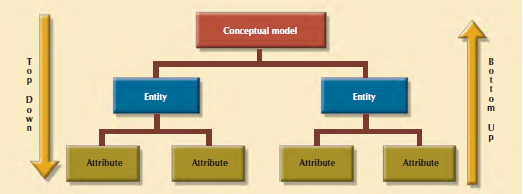
use, performance, security, concurrency control, transaction processing, and third-party support also influenceDBMS software selection.

* *Underlying model*. This can be hierarchical, network, relational, object/relational, or object-oriented. The most suitable and relevant model should be considered.
* *Portability*. A DBMS can be portable/deployed across platforms, systems, and languages.In other words, a good DBMS and Database Application should be platform independent. High portability will be considered.
* *DBMS hardware requirements*. Items to consider include processor(s), RAM, disk space, and so on.

**DATABASE DESIGN STRATEGY**

There are two classical approaches to database design:

* **Top-down design:** In this design approach, the entities are first identified and then their data items (fields/attributes) are then defined. In other words, this strategy starts by identifying the data sets and then defines the data elements for each of those sets.This process involves the identification of different entity types and the definition of each entity’s attributes.
* **Bottom-up design:** In this design approach, the data items (fields/attributes) are first identified and then classify them according to the associated entity. In other words, this approach first identifies the data elements (items) and then groups them together in data sets. Simply put, it first defines attributes, and then groups them to form entities.



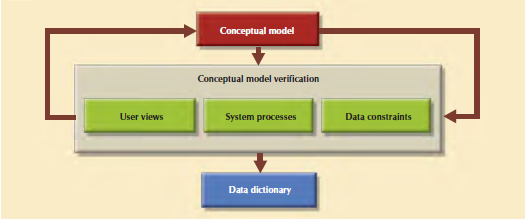
**Top Down:** From **Entity** down (Top DOWN ↓) to the **Attributes**

**Bottom Up:** From **Attributes** up (Bottom-UP ↑) to the **Entity**

Aside the general design approaches (top-down and bottom-up), design approach can also be dependent upon the organizational style or structure. With respect to this, database design strategy can be seen from two perspectives, viz: **Centralized** and **Decentralized Designs.**

**Centralized Design**

This is considered when the design in view is relatively small in size and such that it can be handled by a single individual. Centralized design is considered most suitable when the database design is of relatively small volume and in which the task can be handled by a single designer (database administrator) or bya small, informal design team. The company operations and the scope of the problem are sufficiently limited to alloweven a single designer to define the problem(s), create the conceptual design, verify the conceptual design with the userviews, define system processes and data constraints to ensure the efficacy of the design, and ensure that the designwill comply with all the requirements.



**Decentralized design**

This is the direct opposite of centralized design. This might be used when the data component of the system has a considerable large number of entitiesand complex relations on which very complex operations are performed. Decentralized design is also likely to beemployed when the problem itself is spread across several operational sites and each element is a subset of the entiredata set.

In large and complex projects, the database design typically cannot be done by only one person. Instead, a carefullyselected team of database designers is employed to tackle a complex database project. Within the decentralized designframework, the database design task is divided into several modules. Once the design criteria have been established,the lead designer assigns design subsets or modules to design groups within the team.

