ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**Theory Assignment**

**SUBMITTED BY**

**Pratik Gautam**

**013Bsccsit029**

**4th sem/ 2nd year**

**SUBMITTED TO**

**Er. Sanjaya Kumar Yadav**

**Lecturer**

**Department of Computer Science**

**St. Xavier’s College**

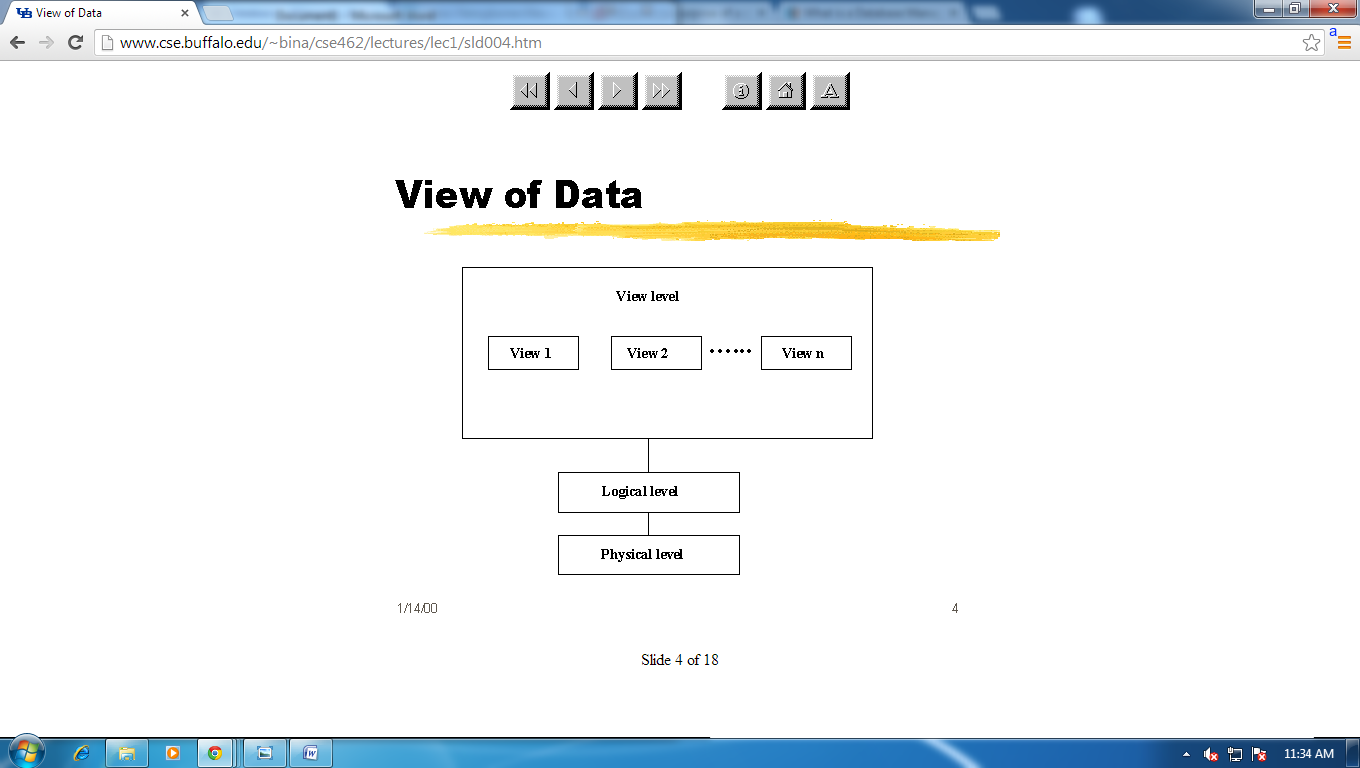
**Date of submission: 2072/ 04/08**

1. **Purpose of database system**

A database is a collection of information that is organized so that it can easily be accessed, managed, and updated. In one view, databases can be classified according to types of content: bibliographic, full-text, numeric, and images. DBMS attempts to solve:

* Data redundancy and inconsistency
* Difficulty in accessing data
* Data isolation multiple files and formats
* Integrity problems
* Atomicity of updates
* Concurrent access by multiple users
* Security problems

1. **View of data**



1. **Database language**

Database language is generic term referring to a class of languages used for defining and accessing databases. A particular database language will be associated with a particular database management system. There are two distinct classes of database language: those that do not provide complete programming facilities and are designed to be used in association with some general-purpose programming language (the host language), and those that do provide complete programming facilities (database programming languages). Some products adopting the former approach seek to minimize host-language programming by the provision of fourth-generation language (4GL) facilities.   
  
A database language must provide for both [logical-schema](http://www.encyclopedia.com/doc/1O11-logicalschema.html) specification and modification (*data* description) and for retrieval and update (data manipulation). In some cases, particularly products derived from the CODASYL network database standard, these aspects are treated distinctly as the data description language (DDL) and the data manipulation language (DML). Modification to the storage is also generally separately provided.

1. **Relational database**

A relational database is a digital database whose organization is based on the relational model of data, as proposed by E.F. Codd in 1970.This model organizes data into one or more tables (or "relations") of rows and columns, with a unique key for each row. Generally, each entity type described in a database has its own table, the rows representing instances of that type of entity and the columns representing values attributed to that instance. Because each row in a table has its own unique key, rows in a table can be linked to rows in other tables by storing the unique key of the row to which it should be linked (where such unique key is known as a "foreign key"). Codd showed that data relationships of arbitrary complexity can be represented using this simple set of concepts.

1. **Database design**

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and view. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).

**Design process**

1. **Determine the purpose of the database** - This helps prepare for the remaining steps.
2. **Find and organize the information required** - Gather all of the types of information to record in the database, such as product name and order number.
3. **Divide the information into tables** - Divide information items into major entities or subjects, such as Products or Orders. Each subject then becomes a table.
4. **Turn information items into columns** - Decide what information needs to be stored in each table. Each item becomes a field, and is displayed as a column in the table. For example, an Employees table might include fields such as Last Name and Hire Date.
5. **Specify primary keys** - Choose each table’s primary key. The primary key is a column, or a set of columns, that is used to uniquely identify each row. An example might be Product ID or Order ID.
6. **Set up the table relationships** - Look at each table and decide how the data in one table is related to the data in other tables. Add fields to tables or create new tables to clarify the relationships, as necessary.
7. **Refine the design** - Analyze the design for errors. Create tables and add a few records of sample data. Check if results come from the tables as expected. Make adjustments to the design, as needed.
8. **Apply the normalization rules** - Apply the data normalization rules to see if tables are structured correctly. Make adjustments to the tables.
9. **Object-base and semi structured database**

**Semi-structured data** is a form of structured data that does not conform with the formal structure of data models associated with relational databases or other forms of data tables, but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data. Therefore, it is also known as self-describing structure.

In semi-structured data, the entities belonging to the same class may have different attributes even though they are grouped together, and the attributes' order is not important.

Semi-structured data is increasingly occurring since the advent of the Internet where full-text documents and databases are not the only forms of data anymore and different applications need a medium for exchanging information. In object-oriented databases, one often finds semi-structured data.

## Pros and Cons of Using a Semi-structured Data Format

### Advantages

* Programmers persisting objects from their application to a database do not need to worry about object-relational impedance mismatch, but can often serialize objects via a light-weight library.
* Support for nested or hierarchical data often simplifies data models representing complex relationships between entities.
* Support for lists of objects simplifies data models by avoiding messy translations of lists into a relational data model.

### Disadvantages

* The traditional relational data model has a popular and ready-made query language, SQL.
* Prone to "garbage in, garbage out"; by removing restraints from the data model, there is less fore-thought that is necessary to operate a data application.

1. **Transaction Management**

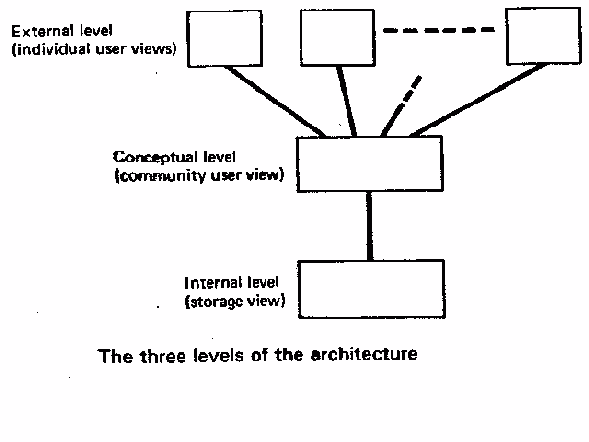
Database transaction is a sequence of actions that are treated as a single unit of work. These actions should either complete entirely or take no effect at all. Transaction management is an important part of and RDBMS oriented enterprise applications to ensure data integrity and consistency. The concept of transactions can be described with following four key properties described as ACID:

* **Atomicity:** A transaction should be treated as a single unit of operation which means either the entire sequence of operations is successful or unsuccessful.
* **Consistency:** This represents the consistency of the referential integrity of the database, unique primary keys in tables etc.
* **Isolation:** There may be many transactions processing with the same data set at the same time, each transaction should be isolated from others to prevent data corruption.
* **Durability:** Once a transaction has completed, the results of this transaction have to be made permanent and cannot be erased from the database due to system failure.

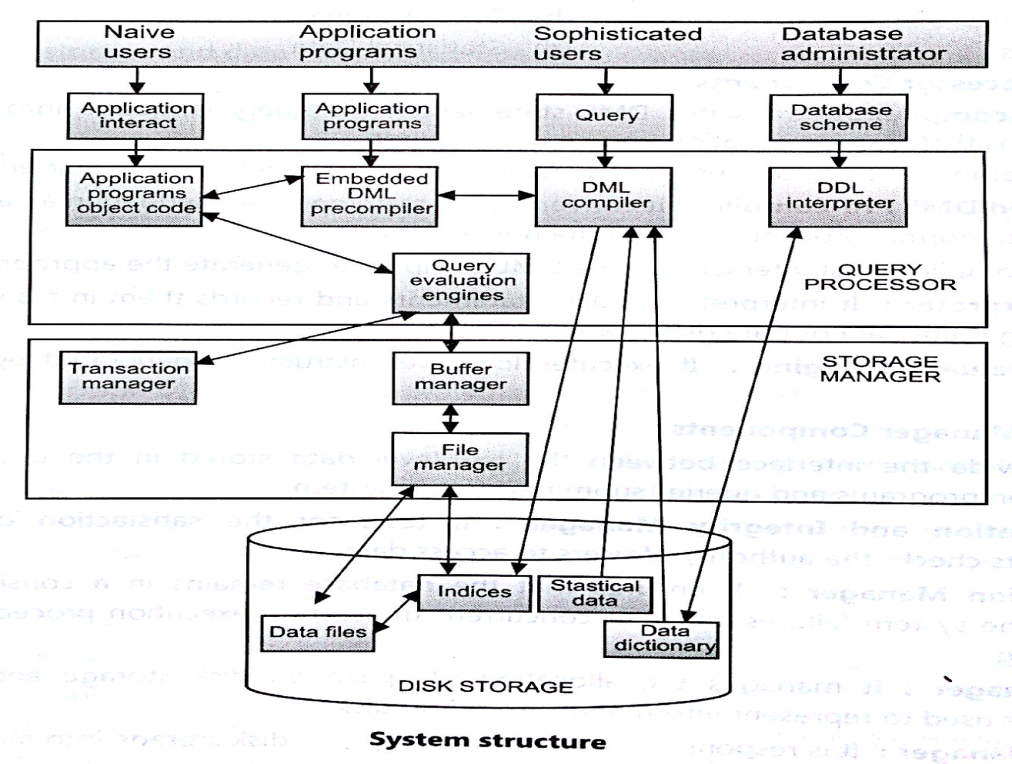
1. **Database Architecture**

In information technology, data architecture is composed of models, policies, rules or standards that govern which data is collected, and how it is stored, arranged, integrated, and put to use in data systems and in organizations. Data is usually one of several architecture domains that form the pillars of an enterprise architecture or solution architecture.

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed, or replaced.



1. **Overall structure**



1. **History of database system**

Following the technology progress in the areas of processors, computer memory, computer storage and computer networks, the sizes, capabilities, and performance of databases and their respective DBMSs have grown in orders of magnitude. The development of database technology can be divided into three eras based on data model or structure: navigational, SQL/relational, and post-relational.

The two main early navigational data models were the hierarchical model, epitomized by IBM's IMS system, and the CODASYL model (network model), implemented in a number of products such as IDMS.

The relational model, first proposed in 1970 by Edgar F. Codd, departed from this tradition by insisting that applications should search for data by content, rather than by following links. The relational model employs sets of ledger-style tables, each used for a different type of entity. Only in the mid-1980s did computing hardware become powerful enough to allow the wide deployment of relational systems (DBMSs plus applications). By the early 1990s, however, relational systems dominated in all large-scale data processing applications, and as of 2015 they remain dominant: Oracle, mySQL and SQL server are the top DBMS. The dominant database language, standardized SQL for the relational model, has influenced database languages for other data models.

Object databases were developed in the 1980s to overcome the inconvenience of object-relational impedance mismatch, which led to the coining of the term "post-relational" and also the development of hybrid object-relational databases.

The next generation of post-relational databases in the late 2000s became known as NoSQL databases, introducing fast key-value stores and document-oriented databases. A competing "next generation" known as NewSQL databases attempted new implementations that retained the relational/SQL model while aiming to match the high performance of NoSQL compared to commercially available relational DBMSs.S