Advanced Database Concepts

A. Components of Database System

1. **Database –** an organized collection of structured information, or data, typically stored electronically in a computer system.
   - A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a **database system** or **database**.
2. **Database Management System (DBMS) –** serves as an interface between the database and its end-users or programs, allowing users to retrieve, update, and manage how the information is organized and optimized.
   - A DBMS also facilitates oversight and control of databases, enabling a variety of administrative operations such as performance monitoring, tuning, and backup and recovery.
3. **Database Application** – a set of one (1) or more computer programs that serves as an intermediary between the user and the DBMS.
   - Application programs read or modify database data by sending database query language to the DBMS.
4. **Users**
   - **Naive users** are unsophisticated users who interact with the system by using predefined user interfaces, such as web or mobile applications.
   - **Application programmers** are computer professionals who write application programs.
   - **Sophisticated users** interact with the system without writing programs. Instead, they form their requests either using a database query language.

Database System Architecture – involves database design and construction on how the databases will operate and function within exisiting structures and location.

a. **Centralized database system –** a database that supports data located at a single site.
b. **Distributed database system** – a collection of multiple interconnected databases, spread physically across various locations.

Structures of Data in DBMS

a. **Structured Data** – data that are stored in relational databases. This type of data uses a predefined and expected format or schema.

ID	Name	Age	Gender	Address	Contact	Email
1	Justin York	27	Male	Makati, PH	+639358491124	justineyork@gmail.com

Table 1*. Structured Data*

b. **Unstructured Data** – data that exist in its original (raw) state that can come in all shapes and sizes.

> The mall show has 200 people attendees. The first attendee to arrive is Justin York. He has an ID number 1, he is male, 27 years old and lives in Makati. His contact # is +639358491124 and email is justineyork@gmail.com.

Table 2*. Unstructured Data*

c. **Semi-structured Data** – individual data items of the same type may have different sets of attributes. It contains semantic tags and there is no pre-defined schema.

```
{
  id: 1,
   Name: "Justin York",
   Age: 27,
   Gender: "Male",
   Address:
   {
          City: "Makati"
          Country: "Philippines
   },
   Contact:
   {
          Mobile: "+639358491124"
          Email: "justineyork@gmail.com"
   }
}
```

Table 3*. Semi-structured Data*

B. Types of Database Management System

- **SQL (Structured Query Language)** – a computer language for storing, manipulating, and retrieving data stored in a relational database.
- **NoSQL**
  - Stands for "Not only SQL" is generally used to describe a new

generation of DBMS that is not based on the traditional relational database model and has been developed to address the challenges represented by Big Data.
  - features a dynamic schema for unstructured data and the data can be stored in different ways.
  - This allows you to create documents without first having to carefully plan and define their structure.
- **NewSQL**
  - Incorporates and builds on the concepts and principles of Structured Query Language (SQL) and NoSQL systems.
  - By combining the reliability of SQL with the speed and performance of NoSQL, NewSQL provides improved functionality and services.
- **Object-Oriented database**
  - A database that stores object rather than data as individual relations. It makes use of object-oriented languages such as *C++, Java, and C#.*
  - Objects can have inheritance relationships with other classes, allowing one object to contain the data of another object as well as the data of the new object inheriting.
  - Choose object-oriented database when you have a business need for high performance on complex data.
- **XML (Extensible Markup Language)**
  - Designed to facilitate the exchange of structured documents.
  - Simplifies data sharing between various systems because of its platform-independent nature.
  - Choose XML as long as your datasets stay relatively small.

C. SQL Extensions

- **PL-SQL** – a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation and stands for Procedural Language.
- **T-SQL –** stands for Transact- SQL. A Microsoft's and Sybase's extension of SQL that adds and declare variables, support transaction control, error and exception handling, and row processing to SQLs existing functions.
- **Other SQL**
  - **PostgreSQL** – an open-source object-relational database system that uses and extends the SQL language.
  - **MySQL** – an open-source SQL relational database management system that was developed and supported by Oracle.
  - **SQLite –** a relational database management system contained in a C programming library.

D. T-SQL Statement Blocks

- **BEGIN/END –** used to define a set of SQL statements that execute together.
- **DECLARE –** used to declare a variable in SQL server. Variable names have to start at "@". If we want to use a variable in SQL server, we need to declare it.
  SET or SELECT – used to assign a value(s) to a variable.
- **CAST –** converts an expression of one data type to another.
- **PRINT –** allows users to return a pre-defined message or value.
  Ex.

```
BEGIN
        DECLARE @FirstVariable AS VARCHAR(100);
        SELECT @FirstVariable = 'My First Variable in SQL';
        PRINT @FirstVariable;
END
```

- **TRY-CATCH –** consists of a try block followed by one or more catch clauses, which specify handlers for different exceptions. This feature helps to deal with any unexpected or exceptional situations that occur when a program/query is running.

Ex.

```
BEGIN
 DECLARE @FirstVariable INT = 0;
 DECLARE @SecondVariable AS VARCHAR(10);

 BEGIN TRY
    SET @SecondVariable = @FirstVariable / 'Error';
 END TRY

 BEGIN CATCH
    PRINT 'Error Message: ' + ERROR MESSAGE();
 END CATCH
END
```

E. Conditional Statements

- **T-SQL Conditional Statements**

IF-ELSE	specify a block of code to be executed, if the returning condition is true
ELSE	specify a block of code to be executed, if the returning condition is false
ELSE-IF	specify a new condition to test, if the previous returning condition is false
WHILE	loops through a block of code as long as a specified condition is true.
CASE	Is the extension of IF-ELSE statement. Unlike IF-ELSE, where only the maximum of one condition is allowed, CASE allows the user to apply multiple conditions to perform different sets of actions

Ex. (While loop)

```
BEGIN
 DECLARE @OrderCapacity INT = 20;
 DECLARE @OrderCount INT = 0;

--Execute a continuous loop until the variable @OrderCapacity is
equal to 0
 WHILE (@OrderCapacity != 0)

  BEGIN
   SET @OrderCount = @OrderCount + 1;
   SET @OrderCapacity = @OrderCapacity - 1;
   PRINT 'Order Count= ' + CAST(@OrderCount AS VARCHAR(10));
   PRINT 'Order Capacity= ' + CAST(@OrderCapacity AS
VARCHAR(10));
  END

PRINT 'TotalOrder= ' + CAST(@OrderCount AS VARCHAR(10));
END
```

Ex. (IF-ELSE AND ELSE-IF CONDITION)

```
BEGIN
DECLARE @v_Grade INT;
DECLARE @v_Result varchar(30);
DECLARE @v_Verdict varchar(10);
SET @v_Grade = 90

--Evaluate if the grade is 75 to 100 then execute the next statement.
IF @v_Grade >= 75 AND @v_Grade <= 100
 BEGIN

  SET @v_Result = 'Excellent!';
  SET @v_Verdict = 'Passed!';

 END
--Evaluate if the grade is 60 to 74 then execute the next statement.
ELSE IF @v_Grade >= 60 AND @v_Grade <= 74
 BEGIN

  SET @v_Result = 'Need improvement.';
  SET @v_Verdict = 'Failed!';

 END
-- If the pre-defined grade can't be evaluated between the IF-ELSE and
ELSE-IF statement then execute the ELSE statement
ELSE

 BEGIN

  SET @v_Verdict = 'Invalid Grade';

 END

PRINT 'Result: ' + @v_Result;
PRINT 'Verdict: ' + @v_Verdict;
PRINT 'Your grade: ' + CAST(@v_Grade AS VARCHAR(20));

END
```

Ex. (CASE Statement)

```
BEGIN
  DECLARE @v_Product INT;
  SET @v_Product = 5;

   SELECT
    CASE @v_Product
    WHEN 1 THEN 'Fruits'
    WHEN 2 THEN 'Vegetables'
    WHEN 3 THEN 'Chicken'
    WHEN 4 THEN 'Beef'
    WHEN 5 THEN 'Noodles'
    ELSE 'No such product.'
   END AS 'Product Category'

END
```

REFERENCES:

Coronel, C. and Morris, S. (2018). *Database systems design, implementation, & management (13th ed.)*. Cengage Learning.

Elmasri, R. & Navathe, S. (2016). *Fundamentals of database systems (7th ed.)*. Pearson Higher Education.

Kroenke, D. & Auer, D. *Database Processing: Fundamentals, design, and implementation (12th ed.)*. Pearson Higher Education.

Silberschatz A., Korth H.F., & Sudarshan, S.(2019). *Database system concepts (7th ed.)*. McGraw-Hill Education.

Microsoft. *Begin-end* and *try-catch statements* (n.d.). Retrieved from https://docs.microsoft.com/

Oracle. *PL-SQL* and *types of databases* (n.d.). Retrieved from https://www.oracle.com/ph

Oracle big data blog. *Structured vs. unstructured data (2019).* Retrieved from https://blogs.oracle.com/bigdata/structured-vs-unstructured-data

PostgreSQL (n.d.). *What is PostgreSQL?* Retrieved from https://www.postgresql.org/

Learn.org (n.d.). *What is database architecture?* Retrieved from https://learn.org/articles/What_is_Database_Architecture.html