**Architecture :**

<https://www.geeksforgeeks.org/dbms-architecture-2-level-3-level/>

**physical files :**

<https://www.geeksforgeeks.org/physical-and-logical-file-systems/>

**Introduction to NoSql Database.**

<https://www.geeksforgeeks.org/introduction-to-nosql/>

**Triggers:**

<https://www.geeksforgeeks.org/sql-trigger-student-database/>

**Cursor management**:

<https://www.geeksforgeeks.org/cursors-in-dbms-definition-types-attributes-uses/>

**parameterized cursors :**

<https://www.geeksforgeeks.org/plsql-parameterized-cursors/>

**Nested Cursors :**

A nested cursor in SQL is a cursor that is defined and used within another cursor. This is typically done in a stored procedure or a script where you need to perform complex data retrieval and processing operations that require looping through results of one cursor while processing results of another.

**Stored procedures :**

<https://www.geeksforgeeks.org/what-is-stored-procedures-in-sql/>

**anonymous block :**

<https://www.geeksforgeeks.org/blocks-in-pl-sql/>

**Privilege and Roles in DBMS:**

<https://www.geeksforgeeks.org/privilege-and-roles-in-dbms/>

**Joins in SQL :**

<https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/>

**SQL LIKE Operator :**

<https://www.geeksforgeeks.org/sql-like/>

**SQL | ALL and ANY :**

<https://www.geeksforgeeks.org/sql-all-and-any/>

**Distributed database:**

<https://www.geeksforgeeks.org/distributed-database-system/>

**Oracle exception handling mechanism :**

<https://docs.oracle.com/database/timesten-18.1/TTPLS/exceptions.htm#TTPLS191>

**branching and looping constructs in ANSI SQL :**

ANSI SQL (American National Standards Institute SQL) is a standardized version of SQL that defines a consistent set of SQL commands and syntax across different relational database management systems (RDBMS). While ANSI SQL itself focuses more on querying and managing databases, many RDBMSs provide procedural extensions to SQL, such as PL/SQL (Oracle), T-SQL (SQL Server), and PL/pgSQL (PostgreSQL), to support more complex logic including branching and looping.

Below are examples of branching and looping constructs in ANSI SQL and its common procedural extensions.

**Branching Constructs**

**1. IF...ELSE**

- Used for conditional branching.

**Example**

```sql

IF (some\_condition)

BEGIN

-- Statements to execute if condition is true

END

ELSE

BEGIN

-- Statements to execute if condition is false

END

```

**Looping Constructs**

**1. LOOP**

- Basic loop structure, typically found in procedural extensions.

**Example**

```sql

DO $$

BEGIN

LOOP

-- Loop body

EXIT WHEN some\_condition; -- Exit condition

END LOOP;

END $$;

```

**2. WHILE Loop**

- Executes as long as the condition is true.

Example (T-SQL - SQL Server)

```sql

WHILE (some\_condition)

BEGIN

-- Loop body

END

```

3. FOR Loop

- Iterates over a range of values or result set.

Example

```sql

DO $$

DECLARE

i INTEGER;

BEGIN

FOR i IN 1..10 LOOP

-- Loop body

END LOOP;

END $$;

```

**User defined functions their limitations.**

User-defined functions are a powerful tool available in SQL, as well as in most programming languages, that allow developers to create their own custom functions. These functions serve specific purposes, such as performing **complex calculations** or **data processing**. They can **accept parameters** and **return values in various forms**, such as a **single scalar quantity** or a **result set**. In this article, we will explore the benefits and usage of user-defined functions in SQL, providing insights into their role in **enhancing query flexibility and efficiency**.

**Types of User-Defined Function in SQL**

There are three types of user-defined functions in SQL:

1. **Scalar Functions:** Scalar functions, also known as **User-Defined Scalar Functions**, are functions that return a single value of a specific data type. These functions are defined in the RETURNS clause. Scalar functions can be categorized as inline or multi-statement functions.
   * **Inline Scalar Functions:** An inline scalar function implies that the returned scalar value is the result of a single statement. It is a concise and efficient way to define a scalar function that performs a calculation or manipulation on input parameters and returns a single value.
   * **Multi-Statement Scalar Functions:** A multi-statement scalar function indicates that the returned scalar value is derived from a series of statements. This type of scalar function allows for more complex logic and calculations, utilizing multiple statements to compute the final scalar value.
2. **Table-Valued Functions:** Table-Valued Functions, also referred to as **User-Defined Table-Valued Functions**, are functions that **return a table as their result**. The output of these functions is typically in the form of a table, similar to the result of a SELECT query. Table-Valued Functions offer the ability to encapsulate complex queries or calculations and return the results as a table. This allows us to treat the function result as a table and perform further operations such as **filtering**, **joining**, or **aggregating** the data.
3. **System Functions:** We could use the SQL built-in system functions to perform a variety of operations. While system functions provide convenience and versatility, one disadvantage is that they cannot be modified or customized according to specific user needs. These functions are predefined by the database management system and their behavior and output are fixed.

**Syntax :**

CREATE FUNCTION new\_function (

@variable DECIMAL(9, 6)

) RETURNS TABLE AS RETURN

SELECT

\*

FROM

new\_table

WHERE

place.variable > @variable;

**Introduction of ANSI SQL :**

ANSI SQL stands for American National Standards Institute Structured Query Language. It is a standard language for relational database management systems (RDBMS) that follows the ANSI/ISO SQL standard.

ANSI SQL defines a set of syntax rules and commands that are used to interact with relational databases. It provides a standardized way to create, retrieve, update, and delete data in a database.

The ANSI SQL standard is maintained and updated by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). It ensures that SQL is consistent across different database platforms and allows applications to be portable between different systems.

ANSI SQL includes various features such as data definition language (DDL) statements for creating and modifying database objects, data manipulation language (DML) statements for querying and modifying data, and data control language (DCL) statements for managing user permissions and security.

Implementations of ANSI SQL may vary among different database vendors, but they all adhere to the basic syntax and functionality defined by the standard. Some popular relational database systems that support ANSI SQL include Oracle, MySQL, Microsoft SQL Server, and PostgreSQL.