**Mohammad Seyfi Marandi - Exercise 4**

**The storage size for variable types in a database**

**Numeric Data Types**

* **TINYINT**: 1 byte
* **SMALLINT**: 2 bytes
* **MEDIUMINT**: 3 bytes
* **INT**: 4 bytes
* **BIGINT**: 8 bytes
* **FLOAT**: 4 bytes
* **DOUBLE**: 8 bytes
* **DECIMAL(M, D)**: Varies, typically each 9 digits require 4 bytes

**String Data Types**

* **CHAR(N)**: N bytes (fixed length)
* **VARCHAR(N)**: Up to N bytes, plus 1 or 2 bytes for length storage (variable length)
* **TEXT**: Up to 65,535 bytes (64 KB)
* **MEDIUMTEXT**: Up to 16,777,215 bytes (16 MB)
* **LONGTEXT**: Up to 4,294,967,295 bytes (4 GB)

**Binary Data Types**

* **BLOB**: Up to 65,535 bytes (64 KB)
* **MEDIUMBLOB**: Up to 16,777,215 bytes (16 MB)
* **LONGBLOB**: Up to 4,294,967,295 bytes (4 GB)

**Date and Time Data Types**

* **DATE**: 3 bytes
* **TIME**: 3 bytes
* **DATETIME**: 8 bytes
* **TIMESTAMP**: 4 bytes
* **YEAR**: 1 byte

**Other Data Types**

* **BOOLEAN**: Typically 1 byte
* **ENUM**: 1 or 2 bytes, depending on the number of values
* **SET**: 1, 2, 3, 4, or 8 bytes, depending on the number of values

**Enumeration, Record, Struct, and Tuple**

1. **Enumeration (Enum)**
   * **Type**: Value type
   * **Storage**: Typically on the stack
2. **Record**
   * **Type**:
     + In some languages (like C# 9.0+), records can be either reference types or value types, depending on their definition (class-based records are reference types, struct-based records are value types).
   * **Storage**:
     + Reference type records are stored on the heap.
     + Value type records are stored on the stack.
3. **Struct**
   * **Type**: Value type
   * **Storage**: Typically on the stack, unless part of a reference type, in which case they are stored on the heap.
4. **Tuple**
   * **Type**:
     + In some languages, tuples are value types (e.g., C#).
     + In other languages, tuples can be reference types (e.g., Python).
   * **Storage**:
     + Value type tuples are stored on the stack.
     + Reference type tuples are stored on the heap.

**Summary**

* **Value Types**: Enumeration, Struct, Value type Records, Value type Tuples.
  + **Storage**: Stack (typically)
* **Reference Types**: Reference type Records, Reference type Tuples.
  + **Storage**: Heap (typically)

**Difference Between Stack and Heap**

**Stack**

1. **Memory Allocation**:
   * Memory is allocated in a last-in, first-out (LIFO) order.
   * Allocation and deallocation are very fast.
2. **Lifetime**:
   * Variables allocated on the stack have a short lifetime and are automatically deallocated when the function call ends.
3. **Usage**:
   * Typically used for local variables and function call management (e.g., function parameters, return addresses).
4. **Size**:
   * Usually smaller in size compared to the heap.
5. **Management**:
   * Managed automatically by the compiler.
6. **Performance**:
   * Faster access compared to the heap due to the LIFO structure and continuous memory allocation.
7. **Fragmentation**:
   * Less prone to fragmentation since memory is managed in a simple linear structure.

**Heap**

1. **Memory Allocation**:
   * Memory is allocated and freed in a random order.
   * Allocation and deallocation can be slower.
2. **Lifetime**:
   * Variables allocated on the heap can have a long lifetime, and the programmer is responsible for deallocating memory.
3. **Usage**:
   * Used for dynamic memory allocation, such as objects and data structures whose size can vary at runtime.
4. **Size**:
   * Generally larger in size compared to the stack.
5. **Management**:
   * Managed manually by the programmer or through garbage collection.
6. **Performance**:
   * Slower access compared to the stack due to more complex memory management and potential fragmentation.
7. **Fragmentation**:
   * More prone to fragmentation because of arbitrary memory allocation and deallocation patterns.