ESTRUCTURAS DE DATOS

NICOLAS ANDRES ROJAS GOMEZ

CENTRO TECNOLOGICO DE LA MANOFACTURA AVANZADA

BASES DE DATOS SQL Y NOSQL

FICHA:3169892

2025

• ¿Qué es un dato?

Un dato es una representación simbólica de una variable que puede ser cuantitativa o cualitativa. Se expresa mediante números, letras o símbolos y, por sí solo, no tiene significado hasta que se organiza y analiza en un contexto específico.

En informática, los datos son la base de cualquier sistema, ya que alimentan programas y algoritmos para generar información útil.

• ¿qué es información?

la información es el conjunto de datos organizados y procesados en sistemas computacionales para que sean comprensibles y útiles. Se almacena, transmite y manipula mediante tecnologías digitales, permitiendo la automatización de procesos y la toma de decisiones eficientes.

• ¿cómo se puede garantizar la disponibilidad de los datos en la empresa?

Para garantizar la disponibilidad de los datos en una empresa, es fundamental contar con copias de seguridad regulares, un plan de recuperación ante desastres. Además, el monitoreo continuo, la optimización del tráfico de red y el uso de almacenamiento en la nube ayudan a prevenir interrupciones. Mantener bases de datos organizadas y eliminar información obsoleta también mejora la eficiencia y el acceso a los datos cuando se necesiten.

• ¿a qué nos referimos con confiabilidad de la información?

La confiabilidad de la información significa que los datos son precisos, completos y consistentes, lo que garantiza que se pueden usar con seguridad para tomar decisiones. Para lograrlo, es importante asegurarse de que los datos sean válidos, representen correctamente la realidad. En empresas y bases de datos, mantener información confiable evita errores y mejora la eficiencia.

• ¿por qué consideras que la integridad de la información es esencial?

La integridad de la información es esencial porque garantiza que los datos sean precisos, completos y confiables en todo momento. Sin ella, las decisiones empresariales, científicas o estratégicas podrían basarse en información errónea, lo que llevaría a problemas graves.

• ¿qué tanto puede impactar una falla de autenticidad en los datos de una organización?

Una falla de autenticidad en los datos puede afectar la confianza, la toma de decisiones, la reputación y la estabilidad financiera de una organización. Puede generar errores estratégicos, problemas legales y costos adicionales.

Elabore una tabla en la cual simbolice las magnitudes siguientes en sistema numérico
decimal y su correspondencia en los sistemas numéricos descritos:

Decimal	Binario	Octal	Hexadecimal
5050	100111011010	11632	13BA
4096	1000000000000	10000	1000
4567	100011101111	10757	11D7
3800	111011011000	7330	ED8
1098	10001001010	2112	44A
1024	10000000000	2000	400
2048	100000000000	4000	800
2512	100111001000	4710	9D8
1970	11110111010	3732	7B2
680	1010101000	1250	2A8
512	1000000000	1000	200
300	100101100	454	12C
256	100000000	400	100
128	10000000	200	80
190	10111110	276	BE
64	1000000	100	40
56	111000	70	38
10	1010	12	A
780	1100001100	1404	30C
912	1110010000	1620	390
360	101101000	550	168
556	1000101100	1054	22C

Decimal	Binario	Octal	Hexadecimal
428	110101100	654	1AC
990	1111101110	1756	3DE
604	1001011100	1114	25C
506	111111010	772	1FA
100	1100100	144	64

¿Qué operaciones aritméticas pueden ejecutarse con los sistemas numéricos?

En los sistemas numéricos, se pueden realizar diversas **operaciones aritméticas**, similares a las del sistema decimal, pero adaptadas a cada base. Algunas de las principales incluyen:

- **Suma**: Se realiza sumando dígitos en cada posición, teniendo en cuenta la base del sistema. En binario, por ejemplo, 1 + 1 = 10.
- **Resta**: Se efectúa restando dígitos con reglas específicas para cada base, utilizando el concepto de préstamo cuando es necesario.
- **Multiplicación**: Se aplica de manera similar a la decimal, pero ajustando los valores según la base. En binario, se usa la lógica de $0 \times 1 = 0$ y $1 \times 1 = 1$.
- **División**: Se ejecuta siguiendo el mismo principio de la división decimal, pero adaptada a la base correspondiente.
- **Conversión entre bases**: Aunque no es una operación aritmética en sí, es fundamental para trabajar con distintos sistemas numéricos.

¿Cómo se hacen conversiones de un sistema a otro: binario a octal, octal a decimal, decimal a hexadecimal, etc.?

1. Binario a Octal

El sistema octal usa base **8**, y el sistema binario usa base **2**. Para convertir de binario a octal:

1. Agrupa los dígitos binarios en grupos de **tres**, comenzando desde la derecha.
2. Convierte cada grupo en su equivalente octal.

Ejemplo: 101101010 (binario)
Agrupamos en tríos: **101 - 101 - 010**
Convertimos cada grupo a octal:

- 101 → 5
- 101 → 5
- 010 → 2
  Resultado en octal: **552**

---

2. Binario a Decimal

El sistema decimal usa base **10**, mientras que el binario usa base **2**. Para convertir de binario a decimal:

1. Multiplica cada dígito binario por **2 elevado a su posición**, comenzando desde la derecha.
2. Suma los valores obtenidos.

Ejemplo: 1101 (binario)
Aplicamos la fórmula:
$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$
(8) + (4) + (0) + (1) = **13**
Resultado en decimal: **13**

---

3. Binario a Hexadecimal

El sistema hexadecimal usa base **16** y el binario usa base **2**. Para convertir de binario a hexadecimal:

1. Agrupa los dígitos binarios en grupos de **cuatro**, comenzando desde la derecha.
2. Convierte cada grupo a su equivalente hexadecimal.

Ejemplo: 101101010 (binario)
Agrupamos en cuartetos: **1011 - 0101 - 0** (agregamos un **0** a la izquierda para completar un grupo)
Convertimos cada grupo a hexadecimal:

- 1011 → B
- 0101 → 5
- 0000 → 0
  Resultado en hexadecimal: **B50**

---

4. Octal a Decimal

El sistema octal usa base **8**, mientras que el decimal usa base **10**. Para convertir de octal a decimal:

1. Multiplica cada dígito octal por **8 elevado a su posición**, empezando desde la derecha.

2. Suma los valores obtenidos.

Ejemplo: 345 (octal)
Aplicamos la fórmula:
$(3 \times 8^2) + (4 \times 8^1) + (5 \times 8^0)$
(192) + (32) + (5) = **229**
Resultado en decimal: **229**

---

5. Octal a Hexadecimal

Para convertir de octal a hexadecimal, primero **convierte el número octal a binario** y luego a hexadecimal.
Ejemplo: 345 (octal)

1. Convierte cada dígito octal a binario:
    - 3 → 011
    - 4 → 100
    - 5 → 101
      Número binario: **011 100 101**
2. Agrupa en cuartetos: **0001 - 1001 - 0101**
3. Convierte a hexadecimal:
    - 0001 → 1
    - 1001 → 9
    - 0101 → 5
      Resultado en hexadecimal: **195**

---

6. Decimal a Hexadecimal

Para convertir de decimal a hexadecimal:

1. **Divide** el número decimal entre **16**.
2. **Registra el residuo**.
3. **Repite** el proceso con el cociente hasta que sea **0**.
4. Los residuos conforman el número hexadecimal (de abajo hacia arriba).

Ejemplo: 229 (decimal)

1. 229 ÷ 16 = 14, residuo **5**

2. 14 ÷ 16 = 0, residuo **E**
   Resultado en hexadecimal: **E5**

¿Qué son: bit, nibble, byte?

Los términos **bit, nibble y byte** son unidades fundamentales en la informática y la representación de datos digitales:

- **Bit**: Es la unidad más pequeña de información en computación. Representa un estado binario, es decir, **0** o **1**.
- **Nibble**: Es un grupo de **4 bits**. Se usa comúnmente en la representación de números en hexadecimal, ya que un nibble puede representar valores de **0 a 15**.
- **Byte**: Es un conjunto de **8 bits**. Es la unidad básica de almacenamiento en computadoras y permite representar **256 valores diferentes** (de 0 a 255).

¿Cuál es la estructura de un registro de memoria?

Un **registro de memoria** es una unidad de almacenamiento dentro de un sistema informático que guarda datos temporalmente para su procesamiento. Su estructura varía según el tipo de memoria y el propósito, pero generalmente incluye los siguientes componentes:

1. **Dirección de memoria**: Indica la ubicación específica del registro dentro del sistema de almacenamiento.
2. **Dato almacenado**: Contiene la información que se guarda en el registro, que puede ser un número, una instrucción o cualquier otro tipo de dato.
3. **Bits de control**: Incluyen señales que determinan el estado del registro, como permisos de lectura/escritura o indicadores de error.
4. **Tamaño del registro**: Define la cantidad de bits que puede almacenar (por ejemplo, 8, 16, 32 o 64 bits).
5. **Tiempo de acceso**: Indica la velocidad con la que se puede leer o escribir en el registro.

¿Cómo se crea un puntero en los lenguajes de programación?

Un **puntero** es una variable que almacena la dirección de memoria de otra variable. Se utilizan en lenguajes como **C, C++ y Python** para manipular datos de manera eficiente.

Cómo crear un puntero en C/C++

En C y C++, los punteros se declaran con el operador `*` y se asignan con `&` (dirección de memoria):

```cpp
#include <iostream>
using namespace std;

int main() {
    int num = 10;  // Variable normal
    int *ptr = &num;  // Puntero que almacena la dirección de num

    cout << "Valor de num: " << num << endl;
    cout << "Dirección de num: " << &num << endl;
    cout << "Valor almacenado en ptr: " << ptr << endl;
    cout << "Valor al que apunta ptr: " << *ptr << endl;

    return 0;
}
```

Aquí, ptr almacena la dirección de **num**, y ***ptr** accede al valor de **num.**

Cómo crear un puntero en Python

Python no tiene punteros explícitos como C/C++, pero usa referencias de objetos:

```python
num = 10
ptr = num  # ptr apunta a la misma referencia que num


print("Valor de num:", num)
print("Valor de ptr:", ptr)
```

Aquí, **ptr** no almacena una dirección de memoria directamente, pero apunta al mismo objeto que **num.**

¿Qué es un sistema de archivos y cuáles son los más utilizados?

Un **sistema de archivos** es la estructura que organiza y gestiona cómo se almacenan, recuperan y manipulan los datos en un dispositivo de almacenamiento, como un disco duro, una memoria USB o una tarjeta SD. Actúa como un intermediario entre el sistema operativo y los archivos, permitiendo que los datos sean accesibles de manera eficiente.

Sistemas de archivos más utilizados

Los sistemas de archivos varían según el sistema operativo y el tipo de almacenamiento. Algunos de los más comunes incluyen:

- **FAT32**: Compatible con casi todos los sistemas operativos, pero con limitaciones en el tamaño de archivos.
- **exFAT**: Mejorado respecto a FAT32, ideal para unidades externas sin restricciones de tamaño de archivo.
- **NTFS**: Predeterminado en Windows, ofrece seguridad, compresión y soporte para archivos grandes.
- **HFS+ y APFS**: Utilizados en macOS, optimizados para dispositivos Apple.
- **ext4**: Común en Linux, sucesor de ext3 y ext2, con mejoras en rendimiento y estabilidad.

¿Cómo se almacenan los datos en un disco duro?

Los datos en un **disco duro** se almacenan mediante un sistema de grabación magnética o electrónica, dependiendo del tipo de disco. Aquí te explico cómo funciona:

1. Discos duros HDD (Hard Disk Drive)

Los HDD utilizan **platos magnéticos** que giran a alta velocidad. La información se almacena en sectores dentro de estos platos y se accede mediante un **brazo mecánico** con un cabezal de lectura/escritura.

- Los datos se graban en forma de **pulsos magnéticos**.
- Se organizan en **pistas, sectores y cilindros**.
- La velocidad de acceso depende de la **rotación del disco** (RPM).

2. Discos SSD (Solid State Drive)

Los SSD no tienen partes móviles y almacenan datos en **chips de memoria flash**.

- Utilizan **memoria NAND**, que permite acceso rápido.
- No requieren movimiento físico, lo que los hace más rápidos y duraderos.
- Son ideales para mejorar el rendimiento de computadoras y servidores.

¿Cómo se define un objeto(tipo) abstracto?

Un **objeto (o tipo) abstracto** se define como una entidad conceptual que encapsula datos (atributos) y comportamientos (operaciones o métodos), independientemente de cómo se implementen internamente. Este concepto es clave en la **Programación Orientada a Objetos (POO)** y en el diseño de software en general.

1. Nombre

Es el identificador del objeto abstracto, usualmente un sustantivo en singular que describe su propósito o naturaleza.

- Ejemplo: CuentaBancaria, Vehículo, Usuario

2. Características (atributos y propiedades)

Son los datos que describen el estado del objeto. Pueden incluir:

- **Atributos**: valores o variables internas (por ejemplo: saldo, nombre, edad).
- **Propiedades**: atributos que tienen comportamientos definidos para acceder o modificarse (por ejemplo: getSaldo() y setSaldo() en lugar de acceder directamente al atributo saldo).

3. Restricciones

Son las condiciones o reglas que deben cumplir los atributos o el comportamiento del objeto. Estas ayudan a garantizar la **integridad de los datos**.

- Ejemplos:
    - El saldo no puede ser negativo.
    - La edad debe ser mayor que 0.
    - Un vehículo no puede estar encendido si no tiene combustible.

4. Lista de Operaciones (métodos)

Son las **acciones internas** que el objeto puede realizar sobre sí mismo. Estas definen su comportamiento.

- Ejemplos:
    - depositar(monto)
    - retirar(monto)
    - encender()
    - acelerar()

5. Lista de Servicios (funciones)

Son funciones que el objeto ofrece **hacia el exterior** (otros objetos o el usuario). En algunos enfoques, estas pueden coincidir con métodos públicos.

- Diferencia principal:
    - **Métodos**: afectan directamente el estado del objeto.
    - **Servicios**: son funciones que proporcionan funcionalidades externas usando el objeto.
- Ejemplos:
    - consultarSaldo() (servicio que devuelve información sin modificar el estado)
    - transferirA(otraCuenta, monto) (servicio compuesto que usa varios métodos internos)

¿Cuáles son los tipos de dato que existen en lenguajes de programación?

Los **tipos de datos** en programación son categorías que definen el tipo de valores que una variable puede almacenar y manipular. Se dividen en varias clases según su naturaleza y uso. Aquí están los más comunes:

1. Tipos de datos primitivos

Son los más básicos y están presentes en casi todos los lenguajes de programación:

- **Enteros (int)**: Representan números sin decimales, como 10, -5, 1000.
- **Punto flotante (float, double)**: Números con decimales, como 3.14, -0.5, 2.718.
- **Booleanos (bool)**: Solo pueden tener dos valores: true (verdadero) o false (falso).
- **Caracter (char)**: Representa un solo carácter, como 'A', '9', '%'.
- **Cadena de texto (string)**: Secuencia de caracteres, como "Hola mundo".

2. Tipos de datos estructurados

Permiten almacenar múltiples valores en una sola variable:

- **Arreglos (array)**: Conjunto de elementos del mismo tipo, como [1, 2, 3, 4].
- **Listas (list)**: Similar a los arreglos, pero más flexibles.
- **Diccionarios (dict, map)**: Almacenan pares clave-valor, como {"nombre": "Juan", "edad": 25}.
- **Tuplas (tuple)**: Listas inmutables, como (3, "rojo", True).

3. Tipos de datos abstractos

Son estructuras más avanzadas utilizadas en programación orientada a objetos:

- **Clases y objetos (class)**: Permiten definir estructuras personalizadas con atributos y métodos.
- **Enumeraciones (enum)**: Conjunto de valores predefinidos, como DíasSemana = {Lunes, Martes, Miércoles}.

4. Tipos de datos especiales

Algunos lenguajes incluyen tipos específicos para ciertas operaciones:

- **Nulo (null, None)**: Representa la ausencia de valor.
- **Punteros (pointer)**: Almacenan direcciones de memoria (usados en C y C++).
- **Funciones (function)**: Se pueden almacenar como variables en lenguajes como Python y JavaScript.

¿Cuáles son los tipos de dato que existen en las bases de datos?

1. Tipos de datos numéricos

Se utilizan para almacenar valores numéricos, ya sean enteros o decimales:

- **Enteros (INT, BIGINT, SMALLINT)**: Números sin decimales.
- **Decimales (FLOAT, DOUBLE, DECIMAL)**: Números con punto flotante o precisión fija.

2. Tipos de datos de texto

Permiten almacenar caracteres y cadenas de texto:

- **CHAR(n)**: Cadena de longitud fija.
- **VARCHAR(n)**: Cadena de longitud variable.
- **TEXT**: Texto extenso sin límite definido.

3. Tipos de datos de fecha y hora

Se usan para registrar fechas y tiempos:

- **DATE**: Solo fecha (YYYY-MM-DD).
- **TIME**: Solo hora (HH:MM:SS).
- **DATETIME / TIMESTAMP**: Fecha y hora combinadas.

4. Tipos de datos booleanos

Representan valores de **verdadero (TRUE)** o **falso (FALSE)**:

- **BOOLEAN**: Se almacena como 1 (verdadero) o 0 (falso).

5. Tipos de datos binarios

Se utilizan para almacenar datos en formato binario, como imágenes o archivos:

- **BLOB**: Datos binarios grandes.
- **BYTEA**: Datos binarios en PostgreSQL.

6. Tipos de datos espaciales

Se usan en bases de datos geográficas:

- **GEOMETRY**: Coordenadas espaciales.
- **POINT, LINESTRING, POLYGON**: Representaciones geométricas.

¿Qué es un Collation en base de datos?

Un Collation en bases de datos es un conjunto de reglas que define cómo se comparan y ordenan los caracteres dentro de una base de datos. Es especialmente importante para determinar la sensibilidad a mayúsculas, acentos y el orden de clasificación de los datos almacenados