1. Elementos (`<xs:element>`)

Define los elementos en el esquema. Se puede especificar el tipo de datos, valores permitidos, y si es obligatorio o no.

Sintaxis básica:

```
<xs:element name="nombre" type="tipo" minOccurs="n" maxOccurs="n" />
```

Ejemplo:

```
<xs:element name="nombre" type="xs:string" minOccurs="1" maxOccurs="1" />
```

---

2. Tipos Simples (`<xs:simpleType>`)

Define restricciones o patrones para un valor simple, como cadenas, números, fechas, etc.

Sintaxis básica:

```
<xs:simpleType name="nombreTipo">
  <xs:restriction base="tipoBase">
    <!-- Restricciones aquí -->
  </xs:restriction>
</xs:simpleType>
```

Ejemplo:

```
<xs:simpleType name="edadValida">
  <xs:restriction base="xs:int">
    <xs:minInclusive value="0" />
    <xs:maxInclusive value="120" />
  </xs:restriction>
</xs:simpleType>
```

---

3. Tipos Complejos (`<xs:complexType>`)

Permite estructurar elementos que contienen otros elementos o atributos.

Sintaxis básica:

```
<xs:complexType name="nombreTipo">
  <xs:sequence>
    <!-- Definición de elementos hijos -->
  </xs:sequence>
  <xs:attribute name="atributo" type="tipo" use="opcionalidad" />
</xs:complexType>
```

Ejemplo:

```
<xs:complexType name="persona">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string" />
    <xs:element name="edad" type="xs:int" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>
```

---

4. Atributos (`<xs:attribute>`)

Define metadatos o propiedades de un elemento.

Sintaxis básica:

```
<xs:attribute name="nombre" type="tipo" use="opcionalidad" />
```

Ejemplo:

```
<xs:attribute name="id" type="xs:string" use="required" />
```

Opcionalidad:

- `use="required"`: El atributo es obligatorio.
- `use="optional"`: El atributo es opcional.

---

5. Grupos de Elementos (`<xs:group>`)

Permite agrupar elementos que se reutilizan en diferentes partes del esquema.

Sintaxis básica:

```
<xs:group name="nombreGrupo">
  <xs:sequence>
    <!-- Elementos del grupo -->
  </xs:sequence>
</xs:group>
```

Ejemplo:

```
<xs:group name="informacionPersonal">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string" />
    <xs:element name="apellido" type="xs:string" />
  </xs:sequence>
</xs:group>
```

---

6. Referencias a Grupos o Tipos (`<xs:ref>` y `type`)

- **Uso de grupos:**

```
<xs:element ref="nombreGrupo" />
```

- **Uso de tipos definidos:**

```
<xs:element name="empleado" type="persona" />
```

---

7. Enumeraciones (`<xs:enumeration>`)

Define valores permitidos para un elemento o atributo.

Ejemplo:

```
<xs:simpleType name="coloresPermitidos">
  <xs:restriction base="xs:string">
    <xs:enumeration value="rojo" />
    <xs:enumeration value="verde" />
```

```
    <xs:enumeration value="azul" />
  </xs:restriction>
</xs:simpleType>
```

---

1. Restricciones para tipos numéricos

Aplicables a tipos como `xs:int`, `xs:decimal`, `xs:float`, etc.

- **`minInclusive` y `maxInclusive`**
  Define el rango mínimo y máximo (incluyendo esos valores).

  ```
  <xs:restriction base="xs:int">
    <xs:minInclusive value="0" />
    <xs:maxInclusive value="100" />
  </xs:restriction>
  ```

- **`minExclusive` y `maxExclusive`**
  Define el rango mínimo y máximo, excluyendo esos valores.

  ```
  <xs:restriction base="xs:decimal">
    <xs:minExclusive value="0.0" />
    <xs:maxExclusive value="100.0" />
  </xs:restriction>
  ```

- **`totalDigits`**
  Especifica el número total de dígitos permitidos.

  ```
  <xs:restriction base="xs:decimal">
    <xs:totalDigits value="5" />
  </xs:restriction>
  ```

- **`fractionDigits`**
  Define el número máximo de dígitos permitidos después del punto decimal.

  ```
  <xs:restriction base="xs:decimal">
    <xs:fractionDigits value="2" />
  </xs:restriction>
  ```

---

2. Restricciones para cadenas de texto

Aplicables a `xs:string`.

- **`length`**
  Define la longitud exacta de la cadena.

  ```
  <xs:restriction base="xs:string">
    <xs:length value="5" />
  </xs:restriction>
  ```

- **`minLength` y `maxLength`**
  Define la longitud mínima y máxima de la cadena.

  ```
  <xs:restriction base="xs:string">
    <xs:minLength value="3" />
    <xs:maxLength value="10" />
  </xs:restriction>
  ```

- **`pattern`**
  Define un patrón con expresiones regulares que la cadena debe cumplir.

```
<xs:restriction base="xs:string">
  <xs:pattern value="[A-Za-z]{3,10}" />
</xs:restriction>
```

  Este patrón permite cadenas de 3 a 10 caracteres que solo contienen letras mayúsculas o minúsculas.

- **`enumeration`**
  Especifica valores permitidos.

```
<xs:restriction base="xs:string">
  <xs:enumeration value="rojo" />
  <xs:enumeration value="verde" />
  <xs:enumeration value="azul" />
</xs:restriction>
```

---

3. Restricciones para tipos de fecha y hora

Aplicables a tipos como `xs:date`, `xs:time`, `xs:dateTime`.

- **`minInclusive`** y **`maxInclusive`**
  Define un rango de fechas o horas permitido.

```
<xs:restriction base="xs:date">
  <xs:minInclusive value="2020-01-01" />
  <xs:maxInclusive value="2025-12-31" />
</xs:restriction>
```

---

4. Restricciones para identificadores únicos

Para atributos o elementos que deben ser únicos.

- **`unique`**
  Define que un elemento debe tener valores únicos dentro de un conjunto.

```
<xs:unique name="uniqueID">
  <xs:selector xpath="empleado" />
  <xs:field xpath="@id" />
</xs:unique>
```

---

5. Restricciones generales

- **`whiteSpace`**
  Controla cómo se manejan los espacios en blanco:
  - `preserve`: No se eliminan espacios.
  - `replace`: Los espacios en blanco se convierten en espacios simples.
  - `collapse`: Los espacios iniciales, finales y consecutivos se eliminan.

```
<xs:restriction base="xs:string">
  <xs:whiteSpace value="collapse" />
</xs:restriction>
```

---

6. Restricciones en valores booleanos

Aplicables a `xs:boolean`.

- Solo acepta los valores `true`, `false`, `1`, o `0`.

---

7. Restricciones combinadas

Puedes combinar varias restricciones dentro de un `<xs:restriction>`.

Ejemplo:

```
<xs:restriction base="xs:string">
  <xs:minLength value="3" />
  <xs:maxLength value="10" />
  <xs:pattern value="[A-Za-z]+" />
</xs:restriction>
```

Este ejemplo define una cadena que:

- Tiene entre 3 y 10 caracteres.
- Contiene solo letras.

---

1. `simpleContent`

Es para cuando tienes un contenido **simple (texto o números)**, pero necesitas **atributos adicionales**.

Ejemplo:

Si quieres definir un precio que tenga el valor numérico y una moneda como atributo.

Definición XSD:

```
<xs:complexType name="Precio">
  <xs:simpleContent>
    <xs:extension base="xs:decimal">
      <xs:attribute name="moneda" type="xs:string" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

En el XML quedaría así:

```
<precio moneda="USD">19.99</precio>
```

En resumen:

- **`simpleContent`** = Contenido simple + atributos adicionales.

---

2. `complexContent`

Es para cuando tienes un elemento con **estructura compleja (hijos, atributos o ambos)** y necesitas extenderlo o modificarlo.

Ejemplo:

Si tienes una persona básica y quieres crear un empleado que también tenga un puesto.

Definición XSD:

```
<xs:complexType name="Persona">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Empleado">
  <xs:complexContent>
    <xs:extension base="Persona">
      <xs:sequence>
        <xs:element name="puesto" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

En el XML quedaría así:

```
<empleado>
  <nombre>Juan</nombre>
  <puesto>Ingeniero</puesto>
</empleado>
```

En resumen:

- **`complexContent`** = Para extender o modificar tipos complejos.

¿Cómo diferenciarlos?

1. **`complexType`**: Para elementos complejos (hijos, atributos o ambos).
2. **`simpleType`**: Para valores simples con reglas o restricciones.
3. **`simpleContent`**: Contenido simple (texto/números) + **atributos**.
4. **`complexContent`**: Estructuras complejas que se extienden o modifican.