



Universidad
de la Ciudad de
Aguascalientes

BASES DE DATOS PARA CIENCIA DE DATOS

Mentes que transforman el mundo

Bases de Datos Semi-estructuradas

Las bases de datos semiestructuradas son sistemas de almacenamiento de datos que no requieren un esquema rígido como las bases de datos tradicionales. A diferencia de las bases de datos estructuradas, donde los datos se almacenan en tablas con filas y columnas predefinidas, las bases de datos semiestructuradas permiten que los datos tengan una estructura más flexible, con jerarquías y relaciones que no necesariamente siguen un formato fijo.

The university has 5600 students.
John's ID is number 1, he is 18 years old and already holds a B.Sc. degree.
David's ID is number 2, he is 31 years old and holds a Ph.D. degree. Robert's ID is number 3, he is 51 years old and also holds the same degree as David, a Ph.D. degree.

```
<University>
  <Student ID="1">
    <Name>John</Name>
    <Age>18</Age>
    <Degree>B.Sc.</Degree>
  </Student>
  <Student ID="2">
    <Name>David</Name>
    <Age>31</Age>
    <Degree>Ph.D. </Degree>
  </Student>
  ....
</University>
```

ID	Name	Age	Degree
1	John	18	B.Sc.
2	David	31	Ph.D.
3	Robert	51	Ph.D.
4	Rick	26	M.Sc.
5	Michael	19	B.Sc.

Structured Data

vs

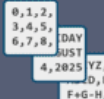
Unstructured Data

Can be displayed
in rows, columns and
relational databases



XY	1	2
A	A1	A2
B	B1	B2
C	C1	C2
D	D1	D2

Numbers, dates
and strings



0, 1, 2,
3, 4, 5,
6, 7, 8,
DAY
JUST
4, 2025
YZ,
D, E
F+G-H,

Estimated 20% of
enterprise data (Gartner)

20%

Requires less storage

==
==
==

Easier to manage
and protect with
legacy solutions



Cannot be displayed
in rows, columns and
relational databases



Images, audio, video,
word processing files,
e-mails, spreadsheets



Estimated 80% of
enterprise data (Gartner)

80%

Requires more storage

==
==
==
==
==
==

More difficult to
manage and protect
with legacy solutions



Extensible Markup Language (XML),
permite definir etiquetas y atributos
para datos no estructurados y
pueden ser almacenados de forma
jerarquica.

```
<studentsList>
  <student id="1">
    <firstName>Greg</firstName>
    <lastName>Dean</lastName>
    <certificate>True</certificate>
    <scores>
      <module1>70</module1>
      <module12>80</module12>
      <module3>90</module3>
    </scores>
  </student>
  <student ind="2">
    <firstName>Wirt</firstName>
    <lastName>Wood</lastName>
    <certificate>True</certificate>
    <scores>
      <module1>80</module1>
      <module12>80.2</module12>
      <module3>80</module3>
    </scores>
  </student>
</studentsList>
```

JavaScript Object Notation (JSON) es una alternativa muy similar al XML pero optimizada para el intercambio en web

```
https://microsoftedge.github.io x +
https://microsoftedge.github.io/Demos/json-dummy-data/256KB-1
[
  {
    "name": "Adeel Solangi",
    "language": "Sindhi",
    "id": "V590F92YF627HFY0",
    "bio": "Donec lobortis eleifend condimentum. Cras dictum d
Maecenas quis nisi nunc. Nam tristique feugiat est vitae mollis. M
    "version": 6.1
  },
  {
    "name": "Afzal Ghaffar",
    "language": "Sindhi",
    "id": "ENTOCR13RSCLZ6KU",
    "bio": "Aliquam sollicitudin ante ligula, eget malesuada n
sem, scelerisque sit amet odio id, cursus tempor urna. Etiam congu
pharetra libero et velit gravida euismod.",
    "version": 1.88
  },
  {
    "name": "Aamir Solangi",
    "language": "Sindhi",
    "id": "IAKPO3R4761JDRVG",
    "bio": "Vestibulum pharetra libero et velit gravida euismo
porttitor sodales ac, lacinia non ex. Fusce eu ultrices elit, vel
    "version": 7.27
  },
  {
    "name": "Abla Dilmurat",
    "language": "Uyghur",
    "id": "5ZVOEPMJUI4MB4EN",
    "bio": "Donec lobortis eleifend condimentum. Morbi ac tell
    "version": 2.53
  },
  {
    "name": "Adil Eli",
```

Hyper-Text Markup Language (HTML) es utilizado para la web,
prove una jeraquia para estructurar datos con etiquetas

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="pingback" href="https://www.datamation.com/xmlrpc.php" />
<meta name='robots' content='index, follow, max-image-preview:large, max-snippet:-1, ma
ink rel="icon" type="image/png" href="https://www.datamation.com/wp-content/uploads/2021/
<!-- This site is optimized with the Yoast SEO plugin v20.6 - https://yoast.com/wordpre
<meta name="description" content="IT & Tech Industry coverage focusing on Emerging
<link rel="canonical" href="https://www.datamation.com/" />
<meta property="og:locale" content="en_US" />
<meta property="og:type" content="website" />
<meta property="og:title" content="Technology News: Latest IT and Tech Industry News" /
<meta property="og:description" content="IT & Tech Industry coverage focusing on Em
<meta property="og:url" content="https://www.datamation.com/" />
<meta property="og:site_name" content="Datamation" />
<meta property="article:modified_time" content="2023-03-08T22:36:57+00:00" />
<meta name="twitter:card" content="summary_large_image" />
<meta name="twitter:label1" content="Est. reading time" />
<meta
```



```
31.286950 15c8 Compile date 2022-10-14 09:07:51
31.287148 15c8 DB SUMMARY
31.287165 15c8 DB Session ID: 44GNK6D01WHC0FED75RW
31.288127 15c8 CURRENT file: CURRENT
31.288147 15c8 IDENTITY file: IDENTITY
31.288271 15c8 MANIFEST file: MANIFEST-000016 size: 531 Bytes
31.288290 15c8 SST files in C:\Users\cb\OneDrive\Pictures\Lightroom\Lightroom Catalog-v13.lrcat-data dir, Total
31.288302 15c8 Write Ahead Log file in C:\Users\cb\OneDrive\Pictures\Lightroom\Lightroom Catalog-v13.lrcat-data:
31.288550 15c8 Options.error_if_exists: 0
31.288556 15c8 Options.create_if_missing: 1
31.288560 15c8 Options.paranoid_checks: 1
31.288564 15c8 Options.flush_verify_memtable_count: 1
31.288568 15c8 Options.track_and_verify_wals_in_manifest: 0
31.288572 15c8 Options.verify_sst_unique_id_in_manifest: 0
31.288576 15c8 Options.env: 0000000022166060
31.288580 15c8 Options.fs: WinFS
31.288585 15c8 Options.info_log: 000000002220A2B0
31.288589 15c8 Options.max_file_opening_threads: 16
31.288592 15c8 Options.statistics: 0000000000000000
31.288596 15c8 Options.use_fsync: 0
31.288600 15c8 Options.max_log_file_size: 0
31.288604 15c8 Options.max_manifest_file_size: 536870912
31.288608 15c8 Options.log_file_time_to_roll: 0
31.288612 15c8 Options.keep_log_file_num: 1000
31.288616 15c8 Options.recycle_log_file_num: 0
31.288620 15c8 Options.allow_fallocate: 1
31.288623 15c8 Options.allow_mmap_reads: 0
31.288627 15c8 Options.allow_mmap_writes: 0
31.288631 15c8 Options.use_direct_reads: 0
31.288635 15c8 Options.use_direct_io_for_flush_and_compaction: 0
31.288639 15c8 Options.create_missing_column_families: 0
31.288642 15c8 Options
31.288646 15c8 Options.wal_dir:
31.288650 15c8 Options.table_cache_numshardbits: 0
31.288654 15c8 Options.WAL_ttl_seconds: 0
31.288658 15c8 Options.WAL_size_limit_MB: 0
31.288661 15c8 Options.max_write_batch_group_size_bytes: 1048576
31.288665 15c8 Options.manifest_preallocation_size: 4194304
```

Log Files

ST*850*1001 ☐
BEG*00*SA*4768*65*050410 ☐
N1*123 MAIN STREET ☐
N4*FAIRVIEW*CA*94618 ☐
PO1*1*100*EA*27.65**VN*331896-42☐
CTT*1*100 ☐
SE*8*1001 ☐

Legend:

ST*Transaction Set ID*Transaction Set Control Number
BEG*Transaction Set Purpose*Purchase Order Date
Number*Release Number*Purchase Order Date
N1*Name Type*Name
N3*Address
N4*City*State*Zip Code
P01*Line Number*Quantity Ordered*Unit of Measure*Product
Basis*Product ID
Qualifier*Product ID
CTT*Number of Line Items*HashTotal
SE*Number of Included Segments*Transaction Set Control

Electronic Data Interchange (EDI) se utiliza ampliamente en diversas industrias, incluidas el comercio, la manufactura, la atención médica, la logística, las finanzas y entre otros, principalmente se utiliza para la digitalización de documentos.

Características

1. Los datos no se rigen por un modelo de datos, pero tienen estructura
2. No es posible almacenar en forma de filas y columnas
3. Contienen metadatos que detallan la estructura en los propios datos
4. Las entidades pueden o no tener los mismo atributos o propiedades

Ventajas

1. Flexibilidad

A diferencia de las bases de datos relacionales, donde se requiere un esquema fijo y predefinido, las bases de datos semiestructuradas permiten almacenar datos sin necesidad de un esquema rígido. Esto es particularmente útil cuando los datos pueden cambiar con el tiempo o cuando se integran datos de múltiples fuentes con diferentes estructuras.

- Facilita la adaptación a cambios en los datos sin necesidad de reestructurar toda la base de datos.

2. Uso de datos jerárquicos o complejos

Los datos que tienen una estructura jerárquica o relaciones complejas (como documentos XML o JSON) se manejan de manera eficiente en bases de datos semiestructuradas.

- Permite representar relaciones complejas y anidadas de manera natural y sencilla.

3. Integración de datos heterogéneos y No Normalizados

Al no requerir un esquema uniforme, es más fácil integrar y manejar datos provenientes de diferentes fuentes, cada una con su propia estructura.

- Mejora la capacidad de fusionar datos de diversas fuentes sin necesidad de normalización o conversión previa.

- Facilita la captura y almacenamiento de datos tal como se encuentran, sin necesidad de un preprocesamiento exhaustivo.

4. Eficiencia en el almacenamiento de datos incompletos (porosidad)

En bases de datos semiestructuradas, no es necesario que todos los registros contengan todas las propiedades, lo que permite almacenar datos incompletos sin generar datos nulos o redundantes.

- Optimiza el almacenamiento y evita desperdicio de espacio en casos donde no toda la información está disponible.

5. Facilidad para manejar cambios en los datos

La estructura flexible permite agregar, eliminar o modificar propiedades de los datos sin necesidad de alterar un esquema fijo.

- Permite a las organizaciones adaptarse rápidamente a nuevos requerimientos sin incurrir en grandes costos de reestructuración.

- Reduce el tiempo y los recursos necesarios para mantener la base de datos cuando hay cambios en los requisitos de los datos.

6. Compatibilidad con Formatos como XML y JSON

Estos formatos son ampliamente utilizados en aplicaciones web y móviles, lo que facilita la integración con tecnologías modernas.

- Mejora la interoperabilidad con aplicaciones y servicios web, y facilita la manipulación de datos en estos formatos.

7. Consulta dinámicas según las necesidades

Aunque puede ser más complicado en comparación con bases de datos relacionales, las consultas en bases de datos semiestructuradas permiten recuperar datos sin necesidad de conocer la estructura completa de los mismos.

- Permite realizar consultas dinámicas y flexibles, adaptándose a las necesidades de la aplicación en tiempo real.

8. Escalabilidad

Muchas bases de datos semiestructuradas, como las bases de datos NoSQL, están diseñadas para escalar horizontalmente, permitiendo manejar grandes volúmenes de datos y solicitudes simultáneas.

- Ofrece la capacidad de crecer junto con las necesidades de la aplicación, manejando grandes volúmenes de datos y tráfico con mayor eficiencia.

Desventajas

1. Integridad de los datos

Es posible que dadas las características de los datos que se almacenen sea complejo aplicar medidas de integridad de datos, así como validación de la información.

- Posibles inconsistencias en los datos que generar errores al momento de explotar la información

2. Rendimiento inferior en ciertos escenarios

Las bases de datos semiestructuradas pueden ser menos eficientes en operaciones que involucren grandes uniones, agregaciones complejas o consultas que requieren explorar múltiples niveles de jerarquía.

- El rendimiento puede degradarse significativamente en comparación con bases de datos relacionales optimizadas para estos tipos de consultas.

3. Menor número de herramientas

- Aunque las herramientas para manejar bases de datos semiestructuradas han avanzado, no siempre son tan maduras o completas como las herramientas disponibles para bases de datos relacionales.

4. Dificultad de análisis

- Se puede complicar el tema del análisis de información en ciertos escenarios dada la naturaleza de la información que puede ser complejo etiquetar o indexar..

Introducción a XML

Introducción a XML

XML (Extensible Markup Language) permite describir y organizar la información de maneras que son fácilmente comprensibles tanto para los seres humanos como para los sistemas. A continuación, puede compartir esa información y su descripción con otros a través de Internet, una extranet, una red o de otras formas.

Es posible utilizar XML para crear su propio lenguaje de marcación que incluya un conjunto de reglas y etiquetas que describan información que se adapte a sus necesidades, por ejemplo, nombre, título, dirección y código postal. Este lenguaje de marcación se define en una definición de tipo de documento (DTD) o un archivo de esquema XML que funciona como la forma estándar de describir la información. El uso de XML para compartir información estandarizada significa que ya no está obligado a escribir programas para centrarse en software propietario o convertir y traducir diferentes formatos de datos.

Aunque tanto XML como HTML utilizan etiquetas para describir el contenido, también son muy diferentes:

1. HTML describe cómo dar formato a la información para su visualización y está destinado a la interacción de equipo a humano.
2. XML describe lo que la información es y está pensada para la interacción de sistema a sistema.

XML utiliza lenguaje humano, no informático. XML es legible y comprensible, incluso por principiantes, y no es más difícil de codificar que HTML.

XML es completamente compatible con lenguajes de programación y 100% portable, cualquier aplicación que pueda procesar XML puede utilizar su información, independientemente de la plataforma.

XML es ampliable, permite crear sus propias etiquetas, o utilizar etiquetas creadas por otros, que utilicen el lenguaje natural de su dominio, que tengan los atributos que necesita y que tengan sentido para usted y sus usuarios.

Introducción a XML

Intercambio de Datos: XML es comúnmente utilizado para intercambiar datos entre sistemas diferentes, ya que es independiente de la plataforma y el lenguaje de programación. Por ejemplo, servicios web (Web Services) suelen usar XML para enviar y recibir datos.

Almacenamiento de Datos: XML puede ser utilizado para almacenar datos estructurados de forma similar a una base de datos, pero en un formato de texto. Esto es útil para configuraciones, datos de aplicaciones y otros tipos de información que necesitan ser fácilmente transportados o leídos por diferentes sistemas.

Estandarización de Documentos: XML es la base para muchos formatos estándar de documentos, como XHTML (una versión de HTML), SVG (para gráficos vectoriales), y muchos otros. Esto permite que los documentos sean compatibles con diferentes herramientas y plataformas.

Configuración de Aplicaciones: Muchas aplicaciones utilizan archivos XML para configurar diferentes parámetros y ajustes. Por ejemplo, archivos de configuración de software como los utilizados en frameworks de desarrollo, servidores web, entre otros.

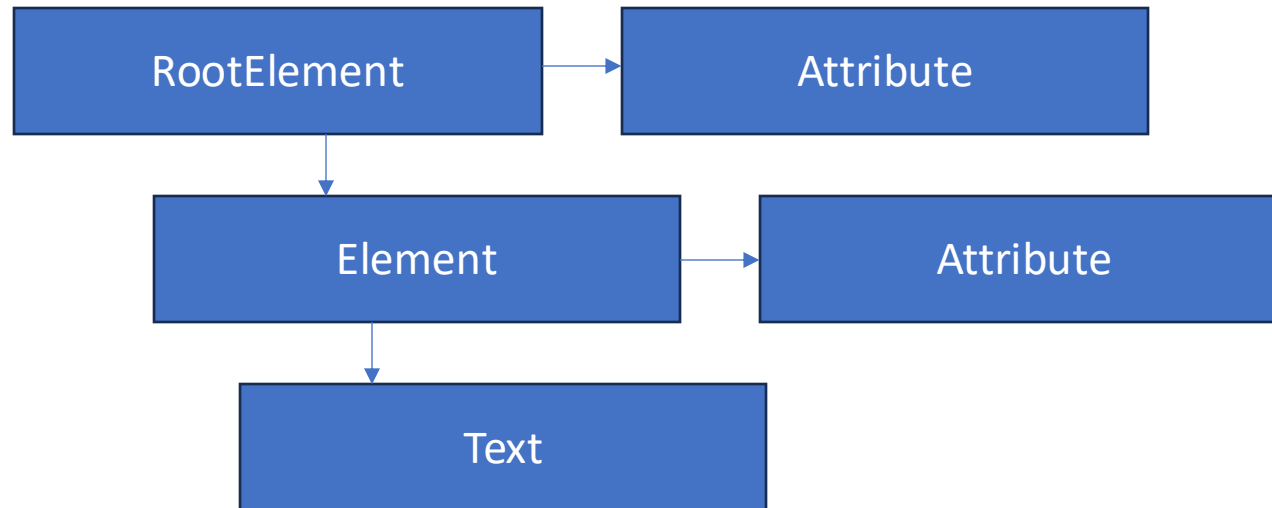
Documentación Técnica: XML es utilizado para la creación de documentación técnica, debido a su capacidad para estructurar información de manera clara y organizada. Ejemplos incluyen DITA (Darwin Information Typing Architecture) para la creación de documentación técnica modular.

Desarrollo de Interfaces de Usuario (UI): En algunos entornos de desarrollo, XML es usado para definir interfaces de usuario, como en Android con los archivos de layout, o en XAML para aplicaciones en .NET.

Bases de Datos XML: Algunas bases de datos están diseñadas específicamente para almacenar y gestionar datos en formato XML, permitiendo consultas y transformaciones de los datos de manera eficiente.

Integración de Aplicaciones: XML facilita la integración entre diferentes aplicaciones y servicios, al proporcionar un formato común para representar datos, lo que es particularmente útil en entornos de empresas donde múltiples sistemas deben interactuar entre sí.

Introducción a XML



Introducción a XML

Element

Un elemento de XML es todo lo que incluya una etiqueta de inicio `<algo>` y una etiqueta de fin `</algo>`

Puede contener text, atributos, otros elementos o una combinación de ambos

Attribute

Información adicional de los elementos que debe ser entrecomillada `<algo atributo="1"></algo>`

```
<library>
  <book>
    <title>XML Basics</title>
    <author>John Doe</author>
    <price>29.99</price>
  </book>
  <book>
    <title>Advanced XML</title>
    <author>Jane Smith</author>
    <price>39.99</price>
  </book>
  <book>
    <title>XQuery in Action</title>
    <author>Mary Johnson</author>
    <price>49.99</price>
  </book>
</library>
```

XPath

XPath (XML Path Language) es un lenguaje utilizado para navegar y seleccionar nodos en un documento XML. Es una herramienta poderosa y flexible que permite extraer, manipular y analizar datos en XML.

1. Selección de Nodos:

Rutas Absolutas y Relativas: XPath permite seleccionar nodos específicos usando rutas absolutas (comenzando desde la raíz) o relativas (comenzando desde el nodo actual).

Ejemplo de una ruta absoluta: `/catalog/book/title` (selecciona el nodo `<title>` dentro de `<book>` en el nivel raíz `<catalog>`).

Ejemplo de una ruta relativa: `book/title` (selecciona todos los nodos `<title>` que son hijos de cualquier nodo `<book>`).

2. Predicados:

Los predicados en XPath se usan para filtrar nodos seleccionados en función de ciertas condiciones.

Ejemplo: `//book[price>30]` selecciona todos los nodos `<book>` cuyo hijo `<price>` tiene un valor mayor a 30.

Ejemplo: `//book[@category='fiction']` selecciona todos los nodos `<book>` que tienen un atributo `category` igual a "fiction".

XPath

XPath (XML Path Language) es un lenguaje utilizado para navegar y seleccionar nodos en un documento XML. Es una herramienta poderosa y flexible que permite extraer, manipular y analizar datos en XML.

3. Ejes en XPath:

Los ejes en XPath definen la relación entre los nodos en el documento XML.

child: Selecciona los hijos directos del nodo actual.

parent: Selecciona el nodo padre del nodo actual.

ancestor: Selecciona todos los ancestros del nodo actual.

descendant: Selecciona todos los descendientes (hijos, nietos, etc.) del nodo actual.

following-sibling: Selecciona todos los nodos hermanos siguientes al nodo actual.

preceding-sibling: Selecciona todos los nodos hermanos anteriores al nodo actual.

Ejemplo: `/library/book/child::title` Seleccionar todos los <title> elementos hijos de los <book> en la biblioteca.

Ejemplo: `/library/book/title[text()='XML Basics']/parent::book` Seleccionar el nodo <book> que es el padre del <title> cuyo valor es "XML Basics".

XQuery

XQuery (XML Query Language) es un lenguaje diseñado para consultar, transformar y manipular datos almacenados en formato XML. Similar en propósito a SQL en bases de datos relacionales, XQuery permite realizar consultas complejas y obtener resultados específicos de documentos XML, lo que lo convierte en una herramienta esencial que utiliza XML como formato principal de datos.

Características Principales de XQuery

Lenguaje Declarativo: XQuery permite especificar qué datos se desean extraer sin tener que detallar cómo extraerlos. Esto lo hace similar a otros lenguajes de consulta como SQL.

Extensión de XPath: XQuery extiende el lenguaje XPath, lo que significa que todas las capacidades de XPath están disponibles en XQuery, junto con funciones adicionales para manipular datos.

Transformación de Datos: XQuery no solo extrae datos, sino que también puede transformar XML en otros formatos, como HTML, JSON, o incluso otro XML.

Manipulación de Secuencias: XQuery trata los resultados como secuencias de elementos, lo que permite realizar operaciones sobre conjuntos de datos, como ordenar, filtrar, y agrupar.

Soporte para Funciones y Variables: XQuery permite definir funciones y variables, lo que facilita la creación de consultas más complejas y reutilizables.

XQuery

Estructura Básica de XQuery

XQuery utiliza una estructura flexible y poderosa que suele incluir:

FLWOR Expressions: Estas son el núcleo de muchas consultas en XQuery y se componen de cinco cláusulas principales:

For: Itera sobre una secuencia de nodos.

Let: Define variables para almacenar valores.

Where: Filtra los resultados.

Order by: Ordena los resultados.

Return: Especifica el valor o los valores que se deben devolver.

Funciones Integradas: XQuery incluye una variedad de funciones integradas para manipular cadenas, realizar cálculos matemáticos, manejar fechas, etc.

XPath: La sintaxis de XPath se utiliza dentro de XQuery para seleccionar nodos específicos en un documento XML.

XQuery

1. Seleccionar Todos los Títulos de Libros

```
for $book in /library/book return $book/title
```

2. Filtrar Libros por Precio

```
for $book in /library/book where $book/price > 30 return $book/title
```

3. Ordenar Libros por Precio

```
for $book in /library/book order by $book/price return $book/title
```

4. Crear un Resumen en Texto Plano

```
for $book in /library/book return concat($book/title, ' by ', $book/author)
```

5. Transformar XML a HTML

```
for $book in /library/book return <div> <h2>{ $book/title }</h2> <p>Author: {  
$book/author }</p> <p>Price: ${ $book/price }</p> </div>
```

Caso de uso



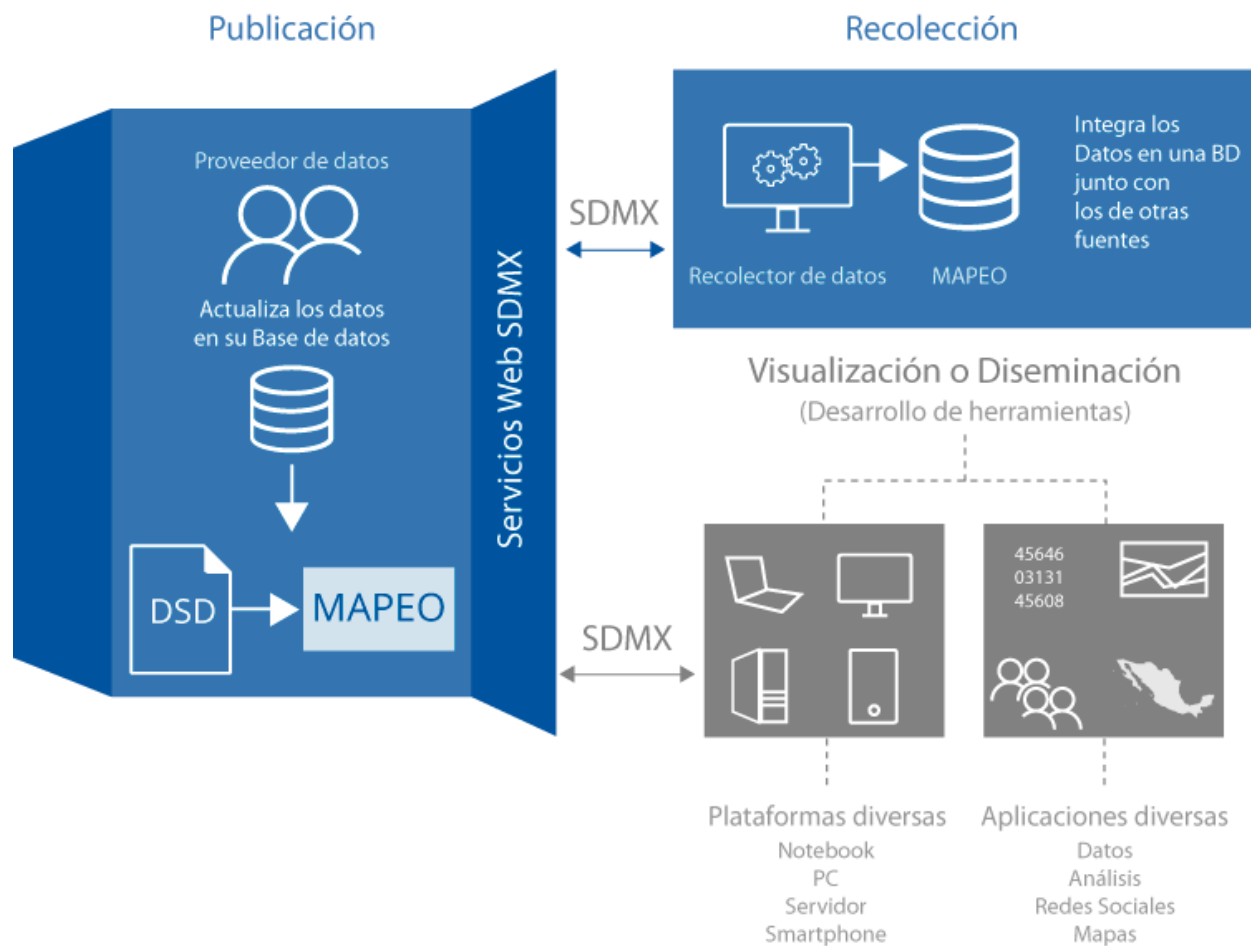
ISO International Standard (ISO 17369).

<https://sdmx.org/>

<https://sdmx.snies.mx/home>

<https://registry.sdmx.org/>

Caso de uso





Universidad
de la Ciudad de
Aguascalientes

Mentes que transforman el mundo

ucags.edu.mx

📞 449 181 2621

📍 Jesús F Contreras #123, Aguascalientes, Mexico, 20070