Dibujo en blanco y negro

Descripción generada automáticamente con confianza media

**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

DIRECCIÓN GENERAL DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN.

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN.

PARTICIPACIÓN DE PROYECTO

*APUNTES Y BITACORA*

Alumno.

*Rodríguez Pichardo Adrián Leonardo.*

Asesor.

Cruz Santiago Pedro Damián

SEPTIEMBRE de 2025 – MARZO de 2026

[APUNTES 3](#_Toc210500009)

[Base de Datos de Series Temporales (TSDB) 3](#_Toc210500010)

[Características de las bases de datos de series temporales 4](#_Toc210500011)

[Optimizado para datos con fecha y hora 4](#_Toc210500012)

[Tasas de ingestión elevadas 4](#_Toc210500013)

[Consultas eficientes para intervalos de tiempo 4](#_Toc210500014)

[Políticas de compresión y conservación de datos 4](#_Toc210500015)

[Software OpenSource 5](#_Toc210500016)

[InfluxDB 5](#_Toc210500017)

[TimescaleDB 5](#_Toc210500018)

[Prometehus 6](#_Toc210500019)

[Agentes a Configurar 6](#_Toc210500020)

[Propios de la TSDB 6](#_Toc210500021)

[Independientes de la TSDB 7](#_Toc210500022)

[Graficadores 7](#_Toc210500023)

[Propios de la TSDB 7](#_Toc210500024)

[Independientes de la TSDB 8](#_Toc210500025)

[Contenedores 8](#_Toc210500026)

[Glances 8](#_Toc210500027)

[OAuth2.0 9](#_Toc210500028)

[Principios de OAuth2.0 9](#_Toc210500029)

[Roles de OAuth2.0 9](#_Toc210500030)

[Tokens de acceso y código de autorización de OAuth 2.0 9](#_Toc210500031)

[¿Cómo funciona OAuth 2.0? 10](#_Toc210500032)

[Tipos de concesión en OAuth 2.0 10](#_Toc210500033)

[Grid UNAM 11](#_Toc210500034)

[BITACORA 12](#_Toc210500035)

# 

# APUNTES

## Base de Datos de Series Temporales (TSDB)

Una base de datos de series temporales proporciona aplicaciones en tiempo real con conjuntos de registros basados en un marco temporal. Una base de datos de serie temporal está optimizada para conjuntos de registros en los que los datos cuentan una historia en orden cronológico. La duración de un conjunto de datos puede ser de unos pocos milisegundos entre sí, o una base de datos de series temporales puede usarse para puntos de datos por hora, diarios, mensuales o anuales.

Una base de datos de serie temporal (TSDB) se ha diseñado específicamente para los conjuntos de registros centrados en un marco temporal. Por ejemplo, es posible que necesite actualizaciones minuto a minuto en una aplicación meteorológica. Una base de datos de series temporales puede almacenar datos por minuto para mostrarlos a los usuarios. Una TSDB puede ser una opción cuando se necesitan consultas más refinadas basadas en el tiempo.

Estas bases de datos suelen estar optimizadas para las lecturas y la creación de nuevos registros. Las TSDB se han creado para recuperar mejor los registros basándose en una marca de tiempo en comparación con otras bases de datos comunes. Los TSDB también son mejores para las aplicaciones en tiempo real en las que los usuarios reciben un conjunto lineal de información basado en la marca de tiempo de cada registro.

El motor de almacenamiento TSDB funciona optimizando las consultas para grandes conjuntos de datos y recuperando datos basándose en marcas de tiempo récord.

Como los registros suelen contener un pequeño número de campos, las consultas se simplifican para los desarrolladores. Un TSDB puede devolver miles de registros, pero el conjunto de registros suele contener solo unos pocos campos. El número limitado de campos y las consultas simplificadas aceleran el rendimiento de las aplicaciones en tiempo real y reducen la sobrecarga en el servidor de bases de datos.

**Link:**

* <https://www.purestorage.com/es/knowledge/what-is-a-time-series-database.html>

-----------------------------------------------------------------------------------------------------------------------------

Las bases de datos de series temporales son bases de datos especializadas diseñadas para gestionar datos organizados e indexados por tiempo. A diferencia de las bases de datos tradicionales, que están optimizadas para el almacenamiento de datos de uso general, las TSDB se centran en el almacenamiento, la consulta y el análisis eficientes de secuencias de puntos de datos con marca temporal.

### Características de las bases de datos de series temporales

#### Optimizado para datos con fecha y hora

En esencia, las TSDB están construidas para manejar datos con marcas de tiempo como atributo fundamental. Cada punto de datos de una TSDB incluye una marca de tiempo, que sirve como índice primario. Esto permite a estas bases de datos almacenar y recuperar eficazmente secuencias ordenadas en el tiempo y proporcionar un acceso rápido a tendencias históricas o acontecimientos recientes.

#### Tasas de ingestión elevadas

Los datos de series temporales suelen generarse a un ritmo rápido (una herramienta de supervisión de servidores que captura métricas del sistema en tiempo real). Las TSDB están optimizadas para estas altas tasas de escritura y pueden ingerir grandes cantidades de datos sin ralentizarse ni perder información.

#### Consultas eficientes para intervalos de tiempo

El análisis de datos de series temporales suele implicar la consulta de intervalos o ventanas temporales concretos, como "las últimas 24 horas" o "este año comparado con el año pasado". Las TSDB se construyen teniendo esto en cuenta, ofreciendo capacidades de consulta especializadas que permiten a los usuarios recuperar rápidamente datos sobre intervalos de tiempo definidos. También admiten agregaciones como medias, sumas o tendencias para ofrecer valiosos análisis sin una lógica de consulta compleja.

#### Políticas de compresión y conservación de datos

Para gestionar la enorme cantidad de datos de series temporales que se generan a lo largo del tiempo, las TSDB utilizan técnicas avanzadas de compresión de datos. Estos métodos reducen los requisitos de almacenamiento al tiempo que preservan el rendimiento de la consulta.

Las TSDB suelen incluir políticas de retención para que los usuarios puedan definir cuánto tiempo deben conservarse los datos. Por ejemplo, un sistema puede conservar los datos detallados del último mes, mientras que reduce el muestreo para los datos más antiguos. El muestreo descendente es el proceso de reducir la granularidad de los datos a lo largo del tiempo. Por ejemplo:

* Las lecturas de temperatura en bruto podrían registrarse cada 10 segundos durante los 7 días más recientes.
* Para los datos más antiguos, el sistema puede reducir la muestra a medias horarias para ahorrar espacio, pero conservando las tendencias históricas.

Link:

- <https://www.datacamp.com/es/blog/time-series-database>

### Software OpenSource

InfluxDB

InfluxDB es una popular base de datos de series temporales de código abierto desarrollada por InfluxData. Se diseñó específicamente para altas tasas de ingestión y consultas eficientes de datos con marca de tiempo, lo que la convierte en una solución habitual para la supervisión del IoT, las métricas DevOps y los análisis en tiempo real.

**Texto

El contenido generado por IA puede ser incorrecto.**

TimescaleDB

TimescaleDB es una extensión de código abierto para PostgreSQL diseñada para combinar la potencia de bases de datos relacionales con la funcionalidad de las series temporales. Te permite aprovechar SQL a la vez que manejas eficazmente los datos con fecha y hora. Esto lo hace especialmente adecuado para casos de uso que requieren integrar datos de series temporales con datos relacionales, como la analítica empresarial o la telemetría IoT.

**Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.**

Prometehus

Prometheus es un sistema de supervisión y alerta con una TSDB incorporada, ampliamente adoptado en DevOps para métricas del sistema en tiempo real, seguimiento del rendimiento y gestión de alertas.

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.**

**Link:**

* <https://www.datacamp.com/es/blog/time-series-database>

### Agentes a Configurar

Los agentes en una Base de Datos de Series Temporales (TSDB) son software especializados que se configuran para recopilar, transformar y enviar datos de series temporales al sistema TSDB, monitoreando la fuente de datos (como aplicaciones, servidores o sensores) y adaptando los datos a un formato que la TSDB pueda entender. Los tipos de configuración de agentes varían según la TSDB específica, pero generalmente implican la definición de qué datos se deben recolectar, el formato de los datos y el destino del envío.

#### Propios de la TSDB

Son aquellos que vienen integrados, recomendados o mantenidos oficialmente por la misma TSDB. Están diseñados para funcionar de forma nativa con ella.

**Ejemplos:**

* InfluxDB → usa Telegraf como agente oficial para recolectar métricas y enviarlas directamente a Influx.
* Prometheus → usa Node Exporter, cAdvisor, o los exporters oficiales para recolectar métricas.
* TimescaleDB → se apoya en Timescale Agent (extensiones para PostgreSQL) o integra colectores específicos.
* OpenTSDB → puede usar su propio colector de métricas (tcollector).

**Características:**

* Están optimizados para esa TSDB en particular.
* Generalmente incluyen integración lista para usarse (sin configuraciones extra de compatibilidad).
* Suelen estar mejor documentados y soportados.

#### Independientes de la TSDB

Son agentes que pueden enviar datos a distintas bases o sistemas de almacenamiento de series de tiempo.

**Ejemplos:**

* Collectd → recolecta métricas del sistema y puede enviarlas a InfluxDB, Graphite, Prometheus, etc.
* StatsD → usado para recolectar métricas de aplicaciones y enviarlas a varias TSDB.
* Fluentd / Logstash → recolectan logs/eventos y pueden enviarlos a múltiples TSDBs o motores de búsqueda (Elastic, Loki, etc.).
* Vector (Datadog, Timber.io) → recolecta métricas y logs, con salida a varias bases o plataformas.

**Características:**

* Son más flexibles porque no dependen de una sola TSDB.
* Requieren configuración extra para adaptar el formato de salida a la TSDB destino.
* Útiles en entornos híbridos donde se usan varias TSDBs o diferentes backends.

### Graficadores

Los graficadores para una base de datos de series temporales (TSDB) son herramientas de visualización y monitoreo que extraen datos de la TSDB para presentarlos en formato gráfico e interactivo, permitiendo el análisis de tendencias y el seguimiento de métricas en tiempo real.

#### Propios de la TSDB

Son nativos u oficiales, diseñados por los mismos creadores de la TSDB o pensados para integrarse sin configuración compleja.

**Ejemplos:**

* InfluxDB → trae su propio graficador web llamado Chronograf (y en versiones nuevas, InfluxDB UI).
* Prometheus → incluye un panel web básico integrado para hacer consultas y gráficas simples con PromQL.
* TimescaleDB → ofrece TimescaleDB Dashboard (basado en PostgreSQL) y extensiones para integración directa con visualizadores SQL.
* OpenTSDB → incluye una interfaz web propia para graficar queries.

**Características:**

* Están listos “out of the box” con la TSDB.
* Simples y con menos curva de aprendizaje.
* Limitados en personalización avanzada (comparado con soluciones externas).

#### Independientes de la TSDB

Son herramientas de visualización externas y universales, que se conectan a múltiples TSDBs y otras fuentes de datos.

**Ejemplos:**

* Grafana → el más popular, soporta InfluxDB, Prometheus, TimescaleDB, OpenTSDB, Elasticsearch, Loki, etc.
* Kibana (aunque más de logs con Elasticsearch, puede usarse con métricas).
* Redash → soporta consultas SQL y múltiples fuentes, incluyendo TSDBs.
* Metabase → orientado a SQL, puede integrarse con TimescaleDB (al ser PostgreSQL).

**Características:**

* Flexibles y potentes, con paneles dinámicos y muchas opciones gráficas.
* Funcionan con múltiples TSDBs (y hasta bases relacionales o NoSQL).
* Requieren configuración inicial (conectar datasource, definir queries).

## Contenedores

Un contenedor es un paquete de software que incluye una aplicación y todo lo necesario para que se ejecute, como código, bibliotecas, herramientas y el entorno de ejecución.

Los contenedores virtualizan el sistema operativo y permiten ejecutar la aplicación en cualquier entorno, ya sea local, en la nube o en un centro de datos, de forma aislada y portátil.

A diferencia de una “máquina virtual”, no simula todo un sistema operativo, sino que comparte el mismo núcleo del equipo, lo que los hace más ligeros y rápidos.

## Glances

Glances es una herramienta de monitorización de sistemas de código abierto, multiplataforma (Linux, macOS, Windows) y basada en Python, que permite supervisar en tiempo real diversos aspectos del sistema como la CPU, memoria, disco, red, procesos, temperaturas y más. Presenta la información de manera dinámica y optimizada en un panel fácil de leer, y puede ser usada en terminal o a través de una interfaz web, permitiendo la monitorización remota en modo cliente/servidor.

## OAuth2.0

OAuth 2.0, significa “Open Authorization” (autorización abierta), es un estándar diseñado para permitir que un sitio web o una aplicación accedan a recursos alojados por otras aplicaciones web en nombre de un usuario. Proporciona acceso consentido y restringe las acciones que la aplicación del cliente puede realizar en los recursos en nombre del usuario, sin compartir nunca las credenciales del usuario.

### Principios de OAuth2.0

OAuth 2.0 es un protocolo de autorización y NO un protocolo de autenticación. Está diseñado principalmente como un medio para conceder acceso a un conjunto de recursos, por ejemplo, API’s remotas o datos de usuario.

Auth 2.0 utiliza tokens de acceso. Un Token de acceso es un dato que representa la autorización para acceder a los recursos en nombre del usuario final. OAuth 2.0 no define un formato específico para los tokens de acceso. Sin embargo, en algunos contextos, se suele utilizar el formato JSON Web Token (JWT). Esto permite a los emisores de tokens incluir datos en el propio token. Además, por razones de seguridad, los tokens de acceso pueden tener una fecha de caducidad.

### Roles de OAuth2.0

Los componentes esenciales de un sistema de OAuth 2.0, son los siguientes:

* Propietario del recurso: El usuario o sistema que posee los recursos protegidos y puede conceder acceso a ellos.
* Cliente: El cliente es el sistema que requiere acceso a los recursos protegidos. Para acceder a los recursos, el cliente debe poseer el token de acceso correspondiente.
* Servidor de autorización: Este servidor recibe las solicitudes de tokens de acceso del cliente y las emite una vez que el propietario del recurso se ha autenticado y ha dado su consentimiento. El servidor de autorización expone dos puntos de conexión: el punto de conexión de autorización, que maneja la autenticación interactiva y el consentimiento del usuario, y el punto de conexión de token, que está involucrado en una interacción de máquina a máquina.
* Servidor de recursos: Un servidor que protege los recursos del usuario y recibe las solicitudes de acceso del cliente. Acepta y valida un token de acceso del cliente y le devuelve los recursos adecuados.

### Tokens de acceso y código de autorización de OAuth 2.0

El servidor de autorización de OAuth 2 no puede devolver directamente un token de acceso después de que el propietario del recurso haya autorizado el acceso. En su lugar, y para mayor seguridad, se puede devolver un Código de Autorización, que se cambia por un token de acceso. Además, el servidor de autorización también puede emitir un Token de actualización con el token de acceso. A diferencia de los tokens de acceso, los tokens de actualización suelen tener un largo plazo de caducidad y pueden cambiarse por nuevos tokens de acceso cuando estos caducan. Dado que los tokens de actualización tienen estas propiedades, deben almacenarse de forma segura por los clientes.

### ¿Cómo funciona OAuth 2.0?

En el nivel más básico, antes de poder utilizar OAuth 2.0, el cliente debe adquirir sus propias credenciales, un id de cliente y un client secret, del servidor de autorización para identificarse y autenticarse al solicitar un token de acceso.

Con OAuth 2.0, las solicitudes de acceso son iniciadas por el cliente, por ejemplo, una aplicación móvil, un sitio web, una aplicación de televisión inteligente, una aplicación de escritorio, etc. La solicitud, el intercambio y la respuesta de los tokens siguen el siguiente flujo general:

1. El cliente solicita autorización (solicitud de autorización) al servidor de autorización, proporcionando el id y el client secret como identificación; también proporciona los ámbitos y un URI de extremo (URI de redireccionamiento) al que enviar el token de acceso o el código de autorización.
2. El servidor de autorización autentica al cliente y verifica que los ámbitos solicitados están permitidos.
3. El propietario del recurso interactúa con el servidor de autorización para conceder el acceso.
4. El servidor de autorización redirige de vuelta al cliente con un código de autorización o un token de acceso, según el tipo de concesión. También puede devolverse un token de actualización.
5. Con el token de acceso, el cliente solicita acceso al recurso desde el servidor de recursos.

### Tipos de concesión en OAuth 2.0

Las concesiones son el conjunto de pasos que un cliente tiene que realizar para obtener la autorización de acceso a los recursos. El marco de autorización proporciona varios tipos de concesión para hacer frente a diferentes escenarios:

* Concesión del Código de Autorización: Este flujo, considerado el más seguro, implica un proceso de dos pasos: el servidor de autorización primero devuelve un código de un solo uso al cliente, quien luego lo intercambia por un token de acceso. Está diseñado para aplicaciones web tradicionales con servidor backend, donde el intercambio se puede realizar de forma segura sin exponer el client\_secret.
* Concesión Implícita: En este flujo simplificado y ahora considerado obsoleto, el token de acceso se devuelve directamente al cliente sin un código intermedio. Se desaconseja su uso porque el token a menudo se transmite en la URL, lo que lo hace vulnerable a fugas y robos, representando un riesgo significativo para la seguridad.
* Concesión del Código de Autorización con PKCE: Esta es una evolución más segura del flujo de código de autorización, que incorpora un "desafío" y un "verificador" criptográfico. Es la opción recomendada para aplicaciones que no pueden almacenar un client\_secret de forma segura, como las aplicaciones de página única (SPA) y las aplicaciones móviles nativas, ya que protege contra ataques de interceptación de código.
* Concesión de Credenciales del Propietario del Recurso: Este flujo requiere que el usuario proporcione sus credenciales directas (nombre de usuario y contraseña) a la aplicación cliente, que luego las intercambia por un token. Su uso se limita a clientes de absoluta confianza, como aplicaciones oficiales del mismo proveedor del servicio, y se emplea cuando la redirección al servidor de autorización no es viable.
* Concesión de Credenciales del Cliente: Utilizado para comunicación entre máquinas (M2M), este flujo permite que una aplicación autentique su propia identidad usando su client\_id y client\_secret para obtener un token de acceso. Es ideal para procesos automatizados, microservicios o trabajos en segundo plano donde no hay un usuario final involucrado que interactúe con la aplicación.
* Flujo de Autorización de Dispositivos: Este flujo está diseñado específicamente para dispositivos con capacidades de entrada limitadas, como televisiones inteligentes o consolas. El dispositivo muestra un código y una URL al usuario, quien luego los introduce en un navegador en otro dispositivo (como su teléfono) para autorizar la aplicación de forma segura.
* Concesión del Token de Actualización: Este no es un flujo de autorización inicial, sino un mecanismo para mantener el acceso. Permite que una aplicación utilice un "token de actualización" de larga duración para obtener un nuevo "token de acceso" cuando el anterior caduca, mejorando la experiencia del usuario al evitar que tenga que volver a iniciar sesión constantemente.

## Grid UNAM

Grid UNAM es una infraestructura distribuida de computación de alto rendimiento / “grid computing” dentro de la UNAM.

Su objetivo es agrupar recursos de cómputo y almacenamiento distribuidos en distintos institutos y dependencias universitarias, para que los proyectos de investigación puedan usar esos recursos de forma compartida y eficiente.

Participan entidades como la DGTIC, LAMOD, el Instituto de Ciencias de la Atmósfera y Cambio Climático, el Instituto de Astronomía, etc.

En cuanto al software se usan herramientas como HTCondor-CE (para interoperar entre nodos) y SLURM (gestión de colas en los nodos) como parte de la arquitectura interna del grid.

Integran sistemas de autenticación / autorización (gestión de identidad) con INDIGO IAM, OpenID / SciTokens, para manejar la seguridad del acceso a recursos distribuidos.

En la parte de usuarios, uno de los requisitos es tener una cuenta en la Red Inalámbrica Universitaria (RIU) que se asocia con INDIGO IAM para generar tokens que permitan el envío de trabajos al grid.

En la documentación “Manejo de tokens” se explica que cuando un trabajo es enviado al sistema de colas, hay un token de acceso que se transporta al “host de ejecución” y se coloca en el entorno de ejecución del trabajo; ese token autoriza al trabajo a acceder a datos durante su ejecución.

En pocas palabras Grid UNAM es una infraestructura de cómputo distribuido con control de acceso, diseñada para que investigadores puedan someter trabajos, hacer cálculos intensivos usando recursos distribuidos, y gestionar acceso seguro a los datos y recursos.

# BITACORA