# Unidad 2

## Bases de datos relacionales

Bases de Datos Aplicada

Universidad Nacional de La Matanza

DIIT
Departamento de Ingeniería e Investigaciones Tecnológicas

# Bases de datos transaccionales

BBDD que tienen como fin el envío y recepción de datos a **gran velocidad**.

Están destinadas generalmente al entorno de análisis de calidad, datos de producción e industrial.

Objetivo principal: asegurar la **consistencia** de las transacciones dentro de una base de datos relacional.

En caso de que no se puedan confirmar, deben ser capaz de **revertirlas**. Evitar que las transacciones queden incompletas, es decir, o se realiza la transacción o no pasa nada (vuelve al estado original).

# Bases de datos transaccionales

***Características***

- Permiten llevar a cabo un **gran número de transacciones** cortas en línea.
- Manejan datos operativos que provienen de sistemas OLTP (on-line transactional processing).
- Capturar datos sobre el **contexto histórico de la transacción** para su posterior análisis.
- Están optimizadas para añadir actualizaciones cortas y rápidas en tiempo real.
- Gran **velocidad** de procesamiento permitiendo realizar consultar y obtener resultados muy rápidamente.

# Bases de datos transaccionales

***Ventajas***

- Permiten asegurar la **integridad** de los datos (puesto que están diseñadas con propiedades ACID).
- Se puede modificar la información sin poner en riesgo dicha integridad.
- Rápidas y operan con muy baja latencia.
- Permiten replicar datos o recuperarlos de los almacenes en muy poco tiempo.

***Desventajas***

- Limitación que tienen para generar informes.
- El historial de datos que facilitan a través de su consulta es limitado a datos actuales o recientes

# Bases de datos transaccionales

**¿Cómo funcionan y para qué sirven?**

- Utilizan lenguaje SQL para acceder y modificar datos dentro de la BBDDs.
- Resultan útiles cuando la integridad de los datos es importante, puesto que no permiten que la transacción se complete si uno de los pasos de la misma falla.
- Cada transacción generará un proceso atómico que puede conllevar bien operaciones de inserción, modificación o borrado de datos. Este proceso debe ser validado con un *commit* o invalidado con *rollback*.

# Bases de datos transaccionales

## Uso y aplicaciones

- Nos permiten obtener datos almacenados de manera muy rápida, y ofrecen una imagen actual de los procesos de la empresa.
- Obtener información para su posterior análisis y toma de decisiones.
- Organizar los datos de las empresas, puesto que permiten dotarles de un esquema común y optimizarlos para el procesamiento de consultas complejas.
- Contextualizar las transacciones llevadas a cabo por aplicaciones operativas.
- Cuando se integran con BBDDs analíticas en una sola plataforma se consigue una mayor consistencia del procesamiento de transacciones

# Bases de datos transaccionales

**Base de datos relacional vs transaccional**

Comprenden dos aspectos de la gestión de datos que van de la mano.

La base de datos **transaccional** funciona de manera asociada a una base de datos **relacional**.

La función de la base de datos transaccional es asegurar que las transacciones dentro de la base de datos relacional se cumplen de manera completa o, en caso de haber un error o fallo en el proceso, se reviertan y por tanto no se completen.

# Bases de datos transaccionales

## Ejemplos

***Transferencias bancarias***: las BBDDs transaccionales se emplean aquí para validar o deshacer la transferencia. Puesto que las transferencias bancarias se desarrollan en dos operaciones distintas, primero se descuenta el dinero en la cuenta origen y luego se suma en la cuenta de destino, si la segunda operación falla, el dinero se devuelve a la cuenta de origen cancelando la operación. El sistema gestor de la base de datos transaccional garantiza que las dos operaciones se cumplen o ninguna de ellas lo haga.

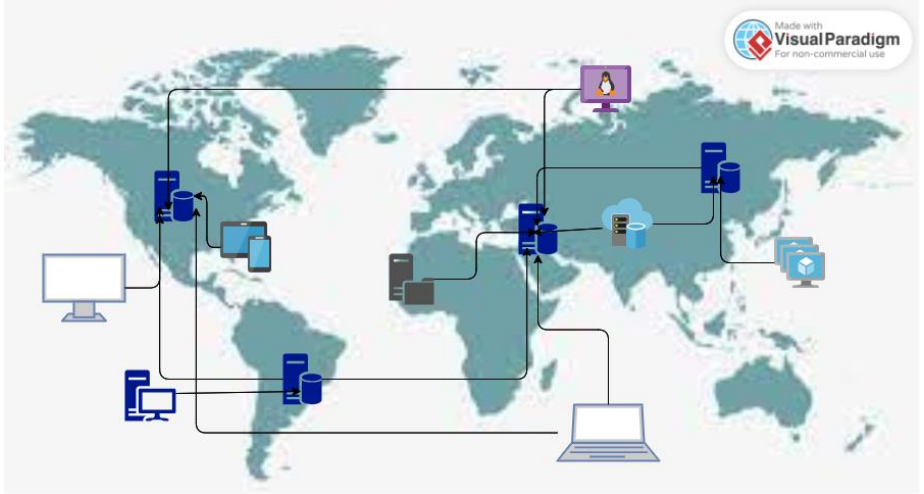
***Sistemas de venta online***: registra una transacción, una venta y un ingreso de dinero para que esta se produzca. Si algún error durante el proceso, la transacción no se completa, y no se produce la venta ni el descuento del dinero.

# Arquitectura de Bases de datos Distribuidas

A nivel arquitectura un sistema distribuido se puede ver como un clúster(1) de servidores

Cuando hablamos de «unir» nos referimos a que comparten recursos de hardware y software; funcionando, así como si fueran un solo sistema unificado.

Es un conjunto de protocolos que permite que varios sistemas de base de datos (clientes, dispositivos, lenguajes) funcionen conjuntamente.




(1)grupo de servidores independientes que trabajan juntos como un solo sistema

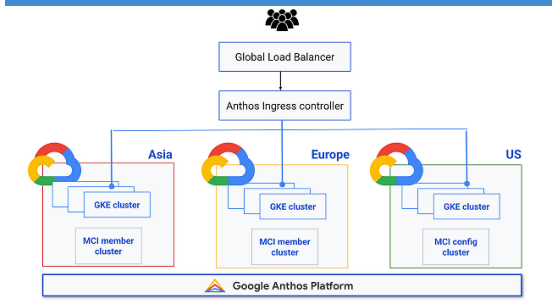
# Arquitectura de Bases de datos Distribuidas

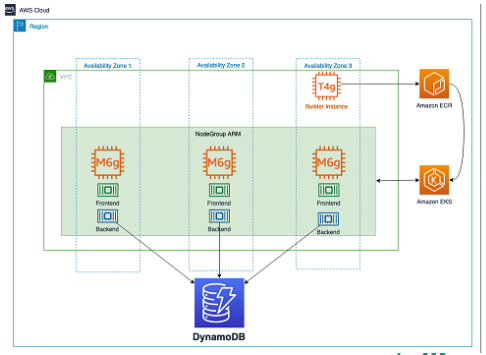
## cluster

• Clúster en la nube: se trata de la configuración y despliegue de clústeres de servidores virtuales o físicos dentro de un entorno de computación en la nube

• Clúster de servidores dedicados: se refiere a un entorno de hosting en el cual un servidor físico es asignado exclusivamente a un solo cliente, organización o para un propósito específico

Los clústeres en la nube pueden desplegarse en nubes públicas, privadas o entornos híbridos, dependiendo de las necesidades de seguridad, regulación y desempeño de la organización.

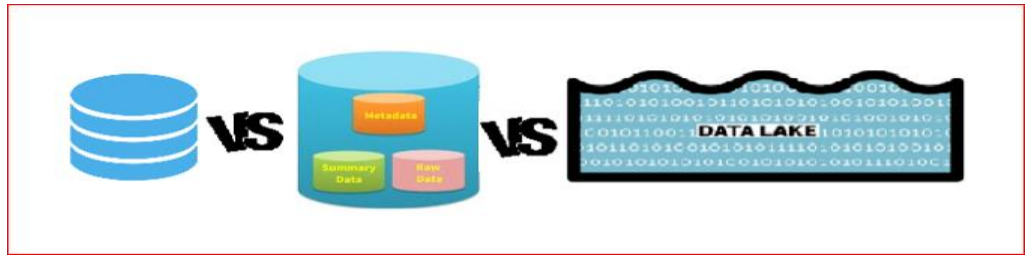






Universidad Nacional de La Matanza
DIIT Departamento de Ingeniería e Investigaciones Tecnológicas

# Base de datos - Data Warehouse – Data Lake

*Son tres conceptos muy **relacionados** entre sí, pero no son lo mismo ni sirven para lo mismo, sirviendo cada uno a un **propósito distinto**.*




## Base de Datos

Los datos se organizan en forma de tablas, filas y columnas. Este tipo de almacenamiento se caracteriza por ser **muy estructurado** y estar **optimizado para realizar operaciones transaccionales** (inserción, borrado, actualización), cuya arquitectura de datos sigue un diseño OLTP (*Online Transaction Processing*). Es el sistema de almacenamiento de datos que se encuentra debajo de los programas informáticos que usamos y que está diseñado para la introducción y modificación de datos en el sistema.

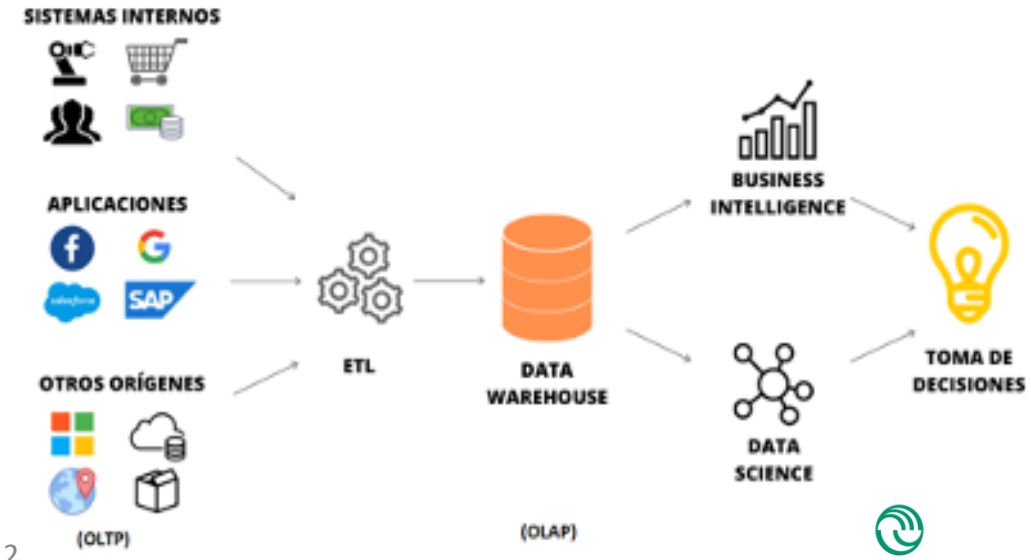
# Base de datos - Data Warehouse – Data Lake



## Data Warehouse (DWH)

Es un sistema de base de datos que se construye encima de todas estas bases de datos **OLAP (Online Analytical Processing)** y que se usa típicamente para ser la fuente de datos de sistemas de **Business Intelligence**. El DWH obtiene la información de todos estos sistemas, limpia los datos, los unifica, les aplica reglas de negocio y finalmente los almacena con una estructura OLAP creando así una capa de datos optimizada para ejecutar análisis de datos sobre ella.

Un DWH es una base de datos con estructura OLAP, que se encuentra **por encima de las bases de datos transaccionales** y que está **diseñada para el análisis** de los datos que hay en el sistema.



# Base de datos - Data Warehouse – Data Lake



## Data Lake (DL)

Es un **repositorio** de datos **centralizado**, para el almacenamiento de datos **estructurados y no estructurados**, en tiempo real o no, donde los datos se almacenan tal cual están en el sistema origen, es decir, sin aplicar ningún tipo de transformación sobre los mismos.

## Similitudes y diferencias entre un DWH y DL

**Diferencias:**

- Un DL almacena datos no estructurados.
- Un DL no transforma los datos, sino que los almacena tal cual le llegan.
- Un DL no almacena los datos con una estructura óptima para su consulta (OLAP).
- Un DL tiene capacidad (de procesamiento y de almacenamiento) para gestionar datos no estructurados en tiempo real.

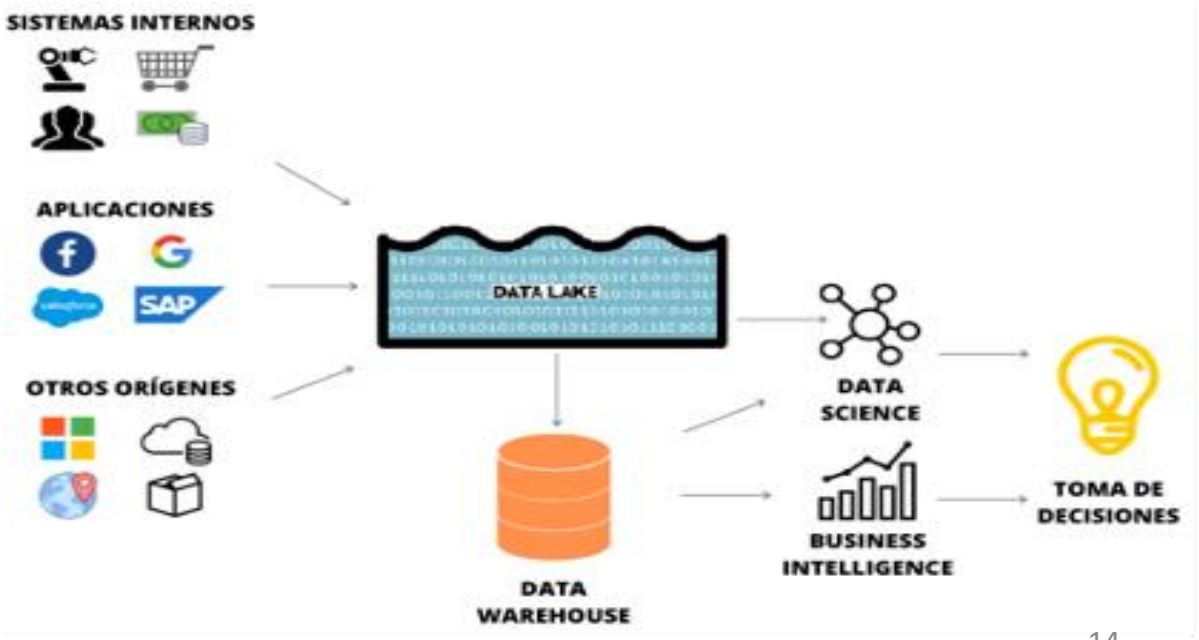
**Similitudes:**

- Son repositorios centralizados de datos.
- Pueden almacenar datos estructurados.
- Pueden gestionar datos estructurados en tiempo real.

# Base de datos - Data Warehouse – Data Lake

## ¿Data Lake o Data Warehouse?

- DWH sirve para el almacenamiento y análisis de datos estructurados.
- DL sirve para el almacenamiento de datos estructurados y no estructurados, aunque no tanto para su análisis.




De esto puede deducirse que **podemos tener un DWH sin un DL, pero rara vez tendremos un DL sin un DWH**.





Universidad Nacional de La Matanza
DIIT Departamento de Ingeniería e Investigaciones Tecnológicas

# Base de datos - Data Warehouse – Data Lake

Si disponemos solamente de datos **estructurados** entonces usaremos un **DWH**.

Si disponemos tanto de datos **estructurados como no estructurados**, entonces:

- Usaremos un **DL** para la **captura** de todo tipo de datos.
- Los datos **estructurados** los procesaremos y organizaremos adecuadamente para su **análisis en un DWH**, siendo el DL la fuente de datos del DWH.
- Los datos **no estructurados se quedarán en el DL**, donde serán accedidos por los procesos que deban analizarlos.

# Motor de base de datos (RDBMS o SGBD)

Proporciona un acceso controlado y un procesamiento de transacciones rápido para cumplir los requisitos de las aplicaciones consumidoras de datos.

***Características opcionales en MS SQL Server***

- Replicación de SQL Server
- Machine Learning Services (R y Python) y extensiones de lenguaje (Java)
- Búsqueda de texto completo
- Data Quality Services
- Servicio de consultas de PolyBase para datos externos: es un componente opcional. A partir de SQL Server 2019, también está disponible el conector de Java para orígenes de datos HDFS.
- Cliente de calidad de datos
- Integration Services
- Componentes de conectividad
- Modelos de programación
- Herramientas de administración
- SQL Server Management Studio (SSMS)
- Componentes de documentación

# Motor de base de datos (RDBMS o SGBD)

Análisis **previo** a la instalación

- ¿Qué hardware se dispondrá para el sistema?
- ¿Se trata de una VM o bare metal? ¿Se alojará en on premises o en la nube?
- ¿Cuánta memoria principal dispone el sistema?
- ¿Cuántos núcleos de CPU tiene el sistema?
- ¿Qué otros servicios proporcionará el mismo sistema? ¿Será también servidor web, fileserver, etc?
- ¿Se utilizará el sistema para aplicación de ciencia de datos?
- ¿Se harán réplicas de alguna DB?
- ¿Qué limitaciones tiene la versión de motor que se está instalando?
- ¿Se harán respaldos en el mismo sistema?
- ¿Cómo se conectarán las aplicaciones cliente (si las hay)?

# Motor de base de datos (RDBMS o SGBD)

**Aspectos clave de la instalación**

- Ubicación de las bases de datos de usuario.
- Ubicación de las bases de datos de sistema.
- Ubicación y parámetro de crecimiento de la base temporal.
- Ubicación de los respaldos.

Planifique cuidadosamente esta configuración considerando:

- Almacenamiento rápido para DBs.
- Almacenamiento *rapidísimo* para la DB temporal.
- Capacidad mayor para respaldos.
- Uso de RAID donde sea posible.
- Evitar el riesgo de llenar la partición de sistema (default).

# Motor de base de datos (RDBMS o SGBD)



## Aspectos clave de la instalación

- Instancia predeterminada o con nombre
  - Un mismo servidor puede alojar más de una instancia.
  - Cada una será un servicio independiente compitiendo por recursos.
- Servicios
  - Active solo los necesarios.
- Intercalación.
  - Seleccione cuidadosamente el collate que servirá para las bases de sistema y default de las nuevas DB.
- Seguridad (contraseñas y métodos de autenticación).
  - Tome en consideración los métodos disponibles en los clientes.

# Motor de base de datos (RDBMS o SGBD)



**Aspectos clave de la instalación**

- Conectividad
  - Cada motor utiliza un puerto TCP por default.
    - No puede haber dos servicios utilizando el mismo puerto.
  - Asegúrese de permitir el tráfico en el firewall local.
- Memoria
  - Por default utiliza la totalidad. Esto puede impactar en el resto del sistema.
- Opciones avanzadas.
  - Mantenga el sistema actualizado.
  - Filestream y características opcionales (Réplica, R, Python, Extracción de texto, etc): instale solo lo necesario.

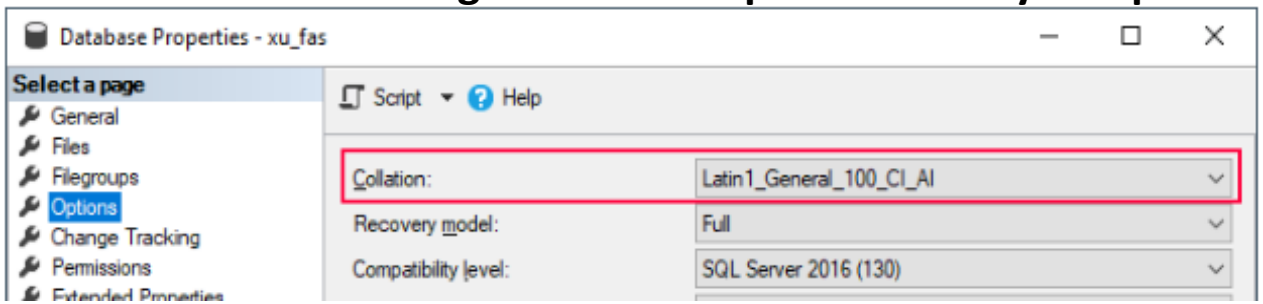
    Menos es más.

# Conexión a un motor de base de datos

- **Paso1: Conexión al Motor de base de datos**
  - *Herramientas de introducción*
  - *Conectarse a Management Studio*
  - *Para conectarse al motor de base de datos*
  - *Autorización de conexiones adicionales*
  - *Crear un inicio de sesión con autenticación de Windows*
- **Paso 2: Conexión desde otro equipo**
  - *Habilitar protocolos*
  - *Cómo habilitar conexiones TCP/IP desde otro equipo*
  - *Configurar un puerto fijo*
  - *Configurar SQL Server para escuchar en un puerto específico*
  - *Abrir puertos del firewall*
  - *Conectarse al motor de base de datos desde otro equipo*
  - *Para conectarse al motor de base de datos desde otro equipo*
  - *Conectarse mediante el Servicio SQL Server Browser*

# Collate - Intercalación

Intercalación (Collation) hace referencia al **patrón de bits utilizado para representar / almacenar cada carácter**, y en consecuencia también se refiere a las **reglas utilizadas para ordenar y comparar** caracteres (Campos de texto).




- Es la propiedad de la BBDDs que proporciona la distinción entre caracteres como acentos, mayúsculas, minúsculas, otros caracteres soportados y reglas de ordenación para los datos que la base de datos es capaz de manejar.
- CS/CI: Case Sensitive Case Insensitive; AS/AI: Accent Sensitive / Accent Insensitive.
- Condicionado por la región donde se vaya a instalar y explotar la base de datos o el aplicativo y la configuración de este parámetro es un punto crítico del despliegue.

# Collate – Intercalación (cadenas de texto)

***Consideraciones particulares***

- La Intercalación de las tablas y objetos de **sistema**.
- La Intercalación de las **nuevas** tablas y objetos que se creen en la base de datos. Si no se especifica de forma explícita la Intercalación deseada en la sentencia CREATE correspondiente, se tomará por defecto la Intercalación de la base de datos.

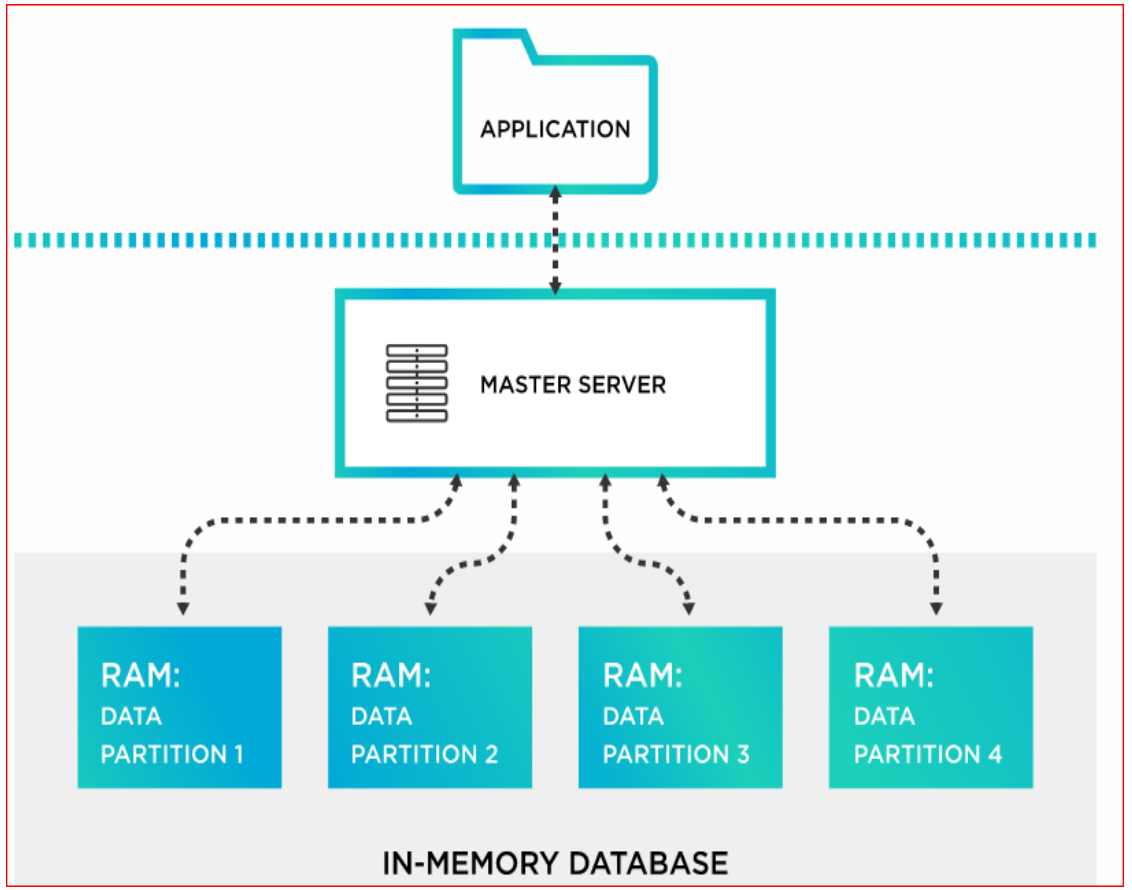
No tener correctamente configurada esta propiedad puede producir comportamientos indeseados. La comparación de cadenas de caracteres puede funcionar de forma distinta a como se pensó o incluso fallar.

Cuando realice **comparaciones** entre campos de texto de distintas DB conviene hacer explícita la intercalación a aplicar, indicándolo como **sufijo** de la misma.

```
WHERE campo1 = campo2 COLLATE Modern_Spanish_CS_AI
```

# Bases de datos en memoria

Almacena todos los datos en la **memoria principal** logrando que el análisis de datos en la BBDDs sea **extremadamente rápido** en comparación con las bases de datos tradicionales que utilizan almacenamiento secundario.



# Bases de datos en memoria

**¿Por qué se necesitan BBDDs en memoria?**

Con la ”internet de las cosas” (IoT) y el crecimiento de las soluciones basadas en la nube, las organizaciones necesitan **procesar datos en tiempo real**. Millones de dispositivos como monitores de salud y seguridad generan datos cada segundo. Los casos más comunes son los siguientes:

- Necesidad de aprovechar las ventajas en tiempo real de Big Data.
- Necesidad de recopilar datos periódicamente y accederlos rápidamente.
- La persistencia de datos no es un gran problema.

# BBDD en memoria vs BBDD en disco

**BBDDs en Memoria**

- Cada operación de lectura/escriturase realiza directamente en memoria (**rápida**).
- Los Datos se pueden perder en una falla de energía (**volátiles**), y deben usar técnicas adicionales para evitar la pérdida de datos.
- Estructura de almacenamiento **simple** ya que el acceso aleatorio es altamente eficiente
- Pueden ejecutarse de manera eficiente en la memoria de los dispositivos integrados
- Ideales para Aplicaciones con **pocos requerimientos de datos.**
- Usadas fuertemente en Aplicaciones basadas en el acceso a internet

**BBDDs en Disco**

- Cada operación requiere lectura/escritura desde almacenamiento masivo, lo que implica una operación de Entrada / Salida (**lento**).
- Los Datos no se pierden con una falla de energía (**persistentes**).
- Estructuras de almacenamientos **complejas**, logrando acceso eficiente al dato.
- Solo pueden ejecutarse en sistemas con dispositivos de almacenamiento secundario.
- Son usadas en Aplicaciones de **grandes volúmenes de datos.**
- Usadas fuertemente en aplicaciones de gestión de clientes y usuarios.

# Bases de datos en memoria

## Implementación de SQL Server

Debemos incluir un `FILEGROUP` en la base de datos que sea para tal fin.

- Podemos hacerlo al crear la DB o modificarla.

Las tablas en memoria se generan de la misma forma que las comunes pero se agrega una cláusula específica.

- Puede además configurarse para que sean persistentes o no.
- Existen algunas consideraciones especiales (p/e no tienen índice clúster).
- Se desempeñan mejor en ambientes de más escrituras que lecturas.

**No deben confundirse con las tablas variables ni con las tablas temporales.**

*Ver guía de "Tablas en memoria" para más ejemplos.*

# Bases de datos en memoria

| Tabla variable | Tabla temporal | Tabla en memoria |
|---|---|---|
| Alcance limitado al módulo (script, SP, trigger) en ejecución. | Alcance a la sesión, excepto temporales globales (mientras se usen). | Alcance idéntico a otras tablas. |
| Intentará mantenerla en memoria (luego va a tempdb). | Se almacenan en tempdb. | Se mantiene en memoria. Respetan el esquema. |
| Se indica en DECLARE con @ como otras variables. | Se indica en CREATE con prefijo # o con doble ## las globales. | Se indica en CREATE con una cláusula, pero el nombre no lo distingue. |
| Su contenido se pierde al quedar fuera de alcance. | Su contenido se pierde al cerrarse la sesión que las creó (o que la usó). | Su contenido no se pierde (salvo en reinicio si no es persistente). |
| No tienen índices (*) ni estadísticas. | Admiten índices y estadísticas. | Admiten índices no clúster. |

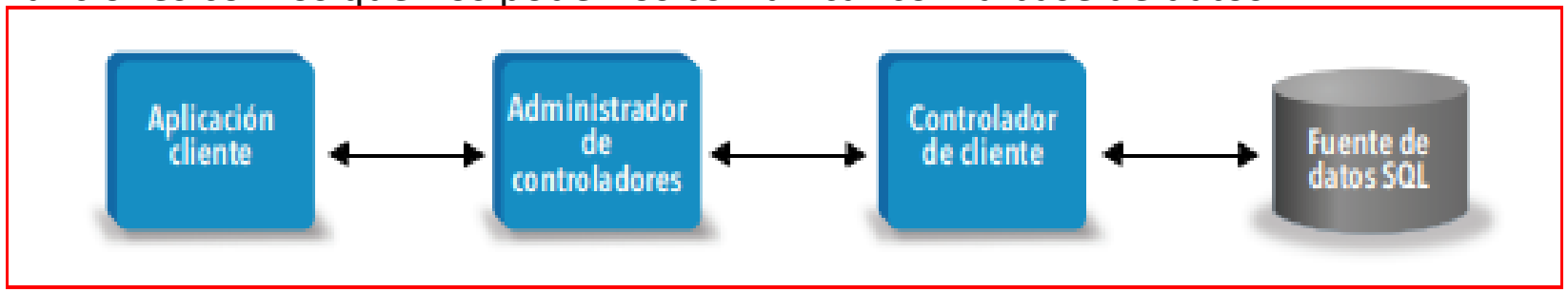
# ODBC y JDBC

ODBC: Open Data Base Connectivity / JDBC: Java Data Base Connectivity

Normas o interfaces de programación de aplicaciones (API).

- ODBC es una API para aplicaciones escritas en el lenguaje C.
- JDBC es una API similar para el lenguaje Java.

Permiten la comunicación con el motor de base de datos. Implementan los protocolos de comunicación necesarios para ejecutar operaciones sobre la base de datos. Exponen una API, es decir una serie de métodos, objetos y funciones con los que nos podemos comunicar con la base de datos.




*Fuente: Silverschatz 4.13.1*

# ODBC y JDBC

Estas API ofrecen a las aplicaciones cliente un lenguaje común para interactuar con diversas fuentes de datos y servicios de base de datos. Todas las aplicaciones compatibles con ODBC y JDBC reconocen un **subconjunto básico de secuencias SQL** (Lenguaje de consulta estructurado).

Si trabaja con SQL, puede utilizar otras aplicaciones (como hojas de cálculo, procesadores de texto y herramientas de generación de informes) para ver, analizar y modificar datos. Mediante las API de ODBC o JDBC, una aplicación cliente se comunica con un administrador de controladores que identifica el controlador de cliente que se va a comunicar con una fuente de datos.

*Fuente: Silverschatz 4.13.1*

# ¿Dudas?

DIIT
Departamento de Ingeniería e Investigaciones Tecnológicas

Universidad Nacional de La Matanza