Alta Disponibilidad de Bases de Datos

Grupo 4

ÍNDICE

1. Definición de alta disponibilidad

La alta disponibilidad en DB se refiere a la capacidad de un sistema de DB para mantener un alto nivel de tiempo de actividad, garantizando que el servicio esté disponible continuamente, prevaleciendo ante fallos de hardware, software o red. Esto se logra mediante la eliminación de puntos únicos de fallo (SPOF), la replicación de datos, la supervisión continua y la capacidad de failover rápido entre nodos activos y de respaldo.

1.1 Diferencia entre HA y Disaster Recovery

La **alta disponibilidad (HA)** se refiere a diseñar sistemas que prolonguen la operación sin interrupciones ante fallos menores como fallos de hardware o red local. En contraste, **Disaster Recovery (DR)** contempla eventos catastróficos mayores: desastres naturales, corrupción masiva de datos. HA maximiza el tiempo de servicio, mientras que DR está pensado para restaurar tras fallos masivos. En muchos diseños se combinan ambas estrategias

1.2 Coste del downtime

El costo del downtime no es solo económico (ventas perdidas, clientes insatisfechos), sino también reputacional y regulatorio (si hay contratos o servicios críticos). Por ejemplo, empresas grandes estiman costos de miles o decenas de miles de dólares por minuto de caída. Esas cifras justifican inversiones elevadas en HA. Además, el nivel de disponibilidad deseada (los “nines”) define cuánto downtime anual se permite, limitando el diseño arquitectónico.

2. Métricas y cómo medir la disponibilidad

2.1 Fiabilidad

La fiabilidad está en relación con la disponibilidad, ya que es la probabilidad de que un sistema desempeñe de forma uniforme la función prevista sin fallos durante un periodo específico. Los equipos deben saber cómo medir y garantizar la fiabilidad para tomar decisiones fundamentales sobre el rendimiento del sistema y mejorar la satisfacción del cliente.

2.2 RTO y RPO

RTO (Recovery Time Objective): Es el tiempo máximo tolerable para restaurar la funcionalidad del sistema después de un fallo.

RPO (Recovery Point Objective): Es cuánto tiempo de datos se puede permitir perder (por ejemplo, si el RPO es 5 minutos, no puede perderse más de los últimos 5 minutos de transacciones).
Estos dos parámetros moldean el diseño: si RTO debe ser muy bajo (segundos), el sistema debe tener failover automatizado y muy confiable; si RPO debe ser cero (sin pérdida), la replicación debe ser sincrónica o muy protegida.

.

3. Patrones arquitectónicos para HA

3.1 Modelo activo-pasivo y multi-master

Activo-Pasivo: Un nodo activo gestiona las operaciones. El nodo pasivo está en espera y toma el control si falla el activo (failover).

Multi–Master: Varios nodos activos aceptan escrituras.

Replicación bidireccional entre nodos

tiene alta disponibilidad y escalabilidad

tiene un mayor riesgo de conflictos y mayor complejidad

3.2 Shared-Storage vs Shared-Nothing

Shared Storage (almacenamiento compartido): Todos los nodos acceden al mismo sistema de almacenamiento
+ Fácil sincronización
- Punto único de fallo si el almacenamiento cae

Shared Nothing (sin almacenamiento compartido): Cada nodo tiene su propio almacenamiento local
Los datos se replican entre nodos
+ Mayor tolerancia a fallos y escalabilidad
- Replicación más compleja

3.3 Geo-replicación y entornos multi-región

Copia de datos entre múltiples regiones para asegurar disponibilidad y rendimiento global.

Tipos:

Sincrónica: actualiza en todas las regiones a la vez.

Asincrónica: replica con retraso.

Ventajas:

- Alta disponibilidad
- Menor latencia
- Recuperación ante fallos

4. Replicación

La **replicación en bases de datos** es un proceso mediante el cual se **duplica y sincroniza la información** de una base de datos entre varios servidores o nodos. En otras palabras, consiste en **mantener copias idénticas de los datos** en distintos lugares para mejorar la **disponibilidad, el rendimiento y la tolerancia a fallos** del sistema.

Cuando un dato se actualiza en una base de datos principal, el sistema de replicación se encarga de **propagar esos cambios** a las demás bases de datos (réplicas) de forma automática o programada, garantizando la coherencia entre todas las copias.

Objetivos principales

- **Disponibilidad:** si un servidor falla, las réplicas pueden seguir operando sin pérdida de datos.
- **Rendimiento:** las consultas pueden repartirse entre varias copias, reduciendo la carga del servidor principal.
- **Tolerancia a fallos:** los sistemas replicados permiten la recuperación rápida ante desastres o caídas.
- **Distribución geográfica:** los usuarios acceden a una copia cercana, mejorando los tiempos de respuesta.
- **Respaldo y seguridad:** las réplicas sirven como copias de seguridad actualizadas.

4.1 Replicación y sus variantes

- **Replicación síncrona**

  Implica que una transacción no se confirma ante el cliente hasta que las réplicas han almacenado los cambios. Esto asegura que no hay pérdida de datos en un fallo repentino (RPO = 0). La desventaja: latencia adicional, especialmente si la réplica está lejos geográficamente.

- **Replicación asíncrona**

  El nodo primario confirma la transacción sin esperar la confirmación de las réplicas. Las réplicas se actualizan posteriormente. La ventaja es rendimiento; la desventaja es riesgo de perder algunos datos si falla el nodo primario antes de que se repliquen esos cambios.

- **Replicación semi-asíncrona**

  Un punto intermedio: la transacción espera que al menos una réplica confirme antes de responder al cliente. Ofrece un balance entre latencia y seguridad de datos.

- **Topologías**

  Maestro-esclavo: un nodo maestro para la escritura, los demás esclavos para la lectura.

  Multi-master: varios nodos pueden escribir.

  En anillo: nodos conectados en ciclo para la replicación ordenada

  Quorum-based: las operaciones de escritura/lectura requieren acuerdo de un conjunto mínimo de réplicas.

5.- Consistencia y teorema CAP

- Consistencia
- Accesibilidad
- Particiones

5.1 Consistencia, disponibilidad

La consistencia garantiza que todos los nodos del sistema muestran los mismos datos al mismo tiempo. Cuando se realiza una lectura o escritura, el cambio debe reflejarse de forma inmediata en todos los nodos. La disponibilidad, en cambio, asegura que el sistema siga respondiendo y accesible, incluso si hay fallas en algunos nodos o problemas en la red.

5.2 Modelos de consistencia

-Consistencia estricta.

-Consistencia linealizable.

-Consistencia Secuencial.

-Consistencia Casual

5.3 Ejemplos de sistemas distribuidos

Hablemos de Google Spanner

Google Spanner es una base de datos distribuida globalmente que ofrece una gran consistencia y escalabilidad horizontal. Utiliza TrueTime, un reloj sincronizado globalmente, para conseguir una coherencia externa.

6. Protocolos

6.1 Paxos y Raft

Objetivo: Garantizar que varios nodos lleguen a un acuerdo (consenso) en sistemas distribuidos.

Paxos: Basado en mensajes entre proposers, acceptors y learners. Muy fiable, pero complejo de implementar.

Raft: Alternativa más sencilla a Paxos. Usa un líder que coordina la replicación de datos. Más fácil de entender y mantener.

6.2 2PC y 3PC

Objetivo: Asegurar que una transacción distribuida se complete en todos los nodos o se deshaga.

2PC (Two-Phase Commit):

1. Fase de preparación (votación).
2. Fase de compromiso o aborto

Simple, pero puede quedar bloqueado si falla el coordinador.

3PC (Three-Phase Commit): Agrega una fase intermedia para evitar bloqueos. Más seguro, pero más complejo y lento.

7. Failover y split-brain

El **failover** es un mecanismo que permite a un sistema **cambiar automáticamente** de un servidor principal (maestro) a un servidor secundario (réplica) cuando el primero deja de funcionar por una caída, error o mantenimiento.
Su objetivo es **mantener la disponibilidad continua del servicio** sin interrupciones para los usuarios.

En un entorno de replicación, el failover se gestiona mediante **monitores o controladores automáticos** que detectan fallos en el nodo principal. Cuando ocurre un problema, estos sistemas designan a una de las réplicas como el **nuevo nodo maestro** y redirigen las conexiones hacia él.
Supongamos que una base de datos MySQL con replicación maestro–esclavo tiene tres nodos.
Si el servidor maestro deja de responder, el sistema de failover promueve automáticamente a uno de los esclavos como nuevo maestro y actualiza las demás réplicas para sincronizarse con este nuevo nodo.

Ventajas del failover
Evita caídas del sistema o interrupciones del servicio.
Aumenta la **tolerancia a fallos** y la disponibilidad.
Reduce la necesidad de intervención manual ante incidentes.

Tecnologías que implementan failover

- **MySQL Replication Manager (MRM)**
- **PostgreSQL Patroni o repmgr**
- **Microsoft SQL Server Always On**
- **Oracle Data Guard**

El **split-brain** (literalmente “cerebro dividido”) es una situación problemática que puede ocurrir en sistemas de replicación o clústeres cuando **dos o más nodos creen simultáneamente ser el servidor principal (maestro)**.
Esto provoca que cada uno acepte escrituras y modificaciones de datos de manera independiente, lo que genera **inconsistencias graves** entre las bases de datos.

Causas comunes del split-brain

- Fallos en la red que interrumpen la comunicación entre nodos.
- Errores en los mecanismos de detección de fallos o en la configuración del failover.
- Pérdida temporal de sincronización o de quorum en el clúster.

Consecuencias

- Inconsistencia de los datos (diferencias entre copias).
- Riesgo de pérdida de información al intentar reconciliar los cambios.
- Posible corrupción de la base de datos si se reestablece la conexión sin control.

7.1 Prevención del split-brain

Para evitar el split-brain, los sistemas distribuidos implementan **mecanismos de control de quorum y comunicación confiable**:

a) Quorum

El quorum define cuántos nodos deben estar de acuerdo antes de aceptar una operación.
Por ejemplo, en un clúster de cinco nodos, puede requerirse que al menos tres estén activos y coordinados para elegir un nuevo maestro.
Esto impide que dos grupos separados del sistema tomen decisiones independientes.

b) Heartbeat (latidos)

Son señales periódicas que los nodos se envían entre sí para confirmar que están activos y conectados.
Si un nodo deja de responder, se activa el protocolo de failover.

c) Fencing o STONITH (“Shoot The Other Node In The Head”)

Mecanismo que **aísla o apaga un nodo problemático** para evitar que siga actuando como maestro y cause conflictos.

d) Monitores externos

Herramientas o servicios externos (como *Pacemaker*, *Corosync* o *Keepalived*) supervisan el estado de los nodos y controlan el failover de forma segura.

8. Operaciones y pruebas de alta disponibilidad

Las operaciones y pruebas de alta disponibilidad (HA) son fundamentales para garantizar que los sistemas de TI mantengan un funcionamiento continuo ante fallos planificados o no planificados, minimizando el tiempo de inactividad y asegurando una experiencia de usuario confiable y consistente.

Estas pruebas se centran en validar que los mecanismos de redundancia, replicación, conmutación por error y recuperación automatizados funcionan correctamente bajo diversas condiciones de fallo

Las pruebas de alta disponibilidad generalmente implican la simulación de fallos controlados en componentes clave del sistema, como servidores, nodos de red o centros de datos.

8.1 Monitoreo y métricas claves

En un entorno centrado en datos, las bases de datos son fundamentales para el rendimiento, la disponibilidad y la escalabilidad. Monitorear métricas clave es vital. Entre ellas destacan:

Tiempo de respuesta de la consulta: mide cuánto tarda en ejecutarse una consulta; valores altos pueden señalar problemas de indexación o ineficiencia.

Transacciones por segundo (TPS): indica el rendimiento y la capacidad de manejar la carga esperada.

Tiempos de espera por bloqueo: reflejan cuánto esperan las consultas por recursos bloqueados; valores altos pueden evidenciar conflictos de concurrencia.

8.2 Pruebas de failover

Cuando una consulta de inserción de errores especifica un bloqueo, fuerza un bloqueo real en la instancia de Aurora PostgreSQL. Otras consultas solo simulan eventos de error sin generarlos realmente. Además, se puede definir la duración de la simulación al enviar la consulta.

8.3 Actualizaciones rolling y blue green

La implementación azul-verde es una estrategia de lanzamiento de aplicaciones que utiliza dos entornos de producción idénticos —denominados “azul” (en vivo) y “verde” (en espera)— para lograr pruebas confiables, lanzamientos sin tiempo de inactividad y reversiones instantáneas. En un momento dado, sólo uno de estos entornos está activo.

Por ejemplo, el entorno azul maneja todo el tráfico de producción en vivo mientras que el entorno verde permanece inactivo. Cuando una nueva versión de la aplicación está lista, se implementa en el entorno verde para realizar pruebas. Una vez que la versión pasa las pruebas, el tráfico se redirige sin problemas de azul a verde.

9. Copias de Seguridad

Garantizan la recuperación de datos ante fallos, errores o pérdidas.

9.1 Backups Físicos vs Lógicos

Físicos: copian directamente los archivos de datos del sistema. Más rápidos, pero dependientes del motor y versión.

Lógicos: exportan la información en formato legible (SQL, JSON...). Más portables, pero más lentos.

9.2 PITR (Point-In-Time Recovery)

Permite restaurar la base de datos a un momento exacto en el tiempo. Usa un backup completo. Evita pérdidas tras errores.

9.3 Herramientas de Backup y Restore

MySQL: mysqldump, mysqlpump, xtrabackup.

PostgreSQL: pg_dump, pg_basebackup, pg_restore.

MongoDB: mongodump, mongorestore.

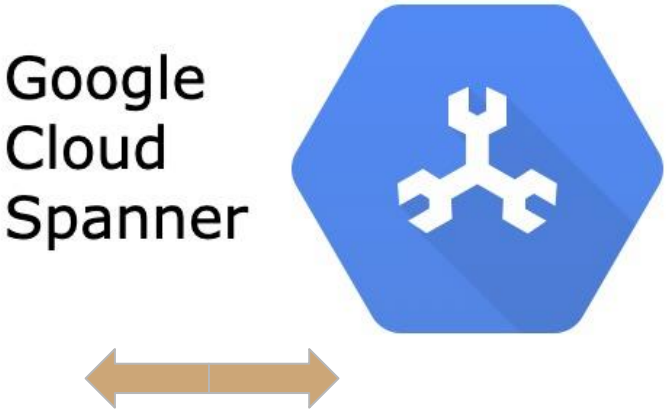
Cloud: snapshots automáticos (AWS RDS, Azure, GCP).

Preguntas

¿Qué es la alta disponibilidad?

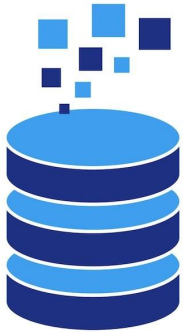
¿Qué es Google Spanner?

¿Cuales son las diferencias entre un Backup físico y lógico?



¿Qué es la alta disponibilidad?

La alta disponibilidad en DB se refiere a la capacidad de un sistema de DB para mantener un alto nivel de tiempo de actividad, garantizando que el servicio esté disponible continuamente, prevaleciendo ante fallos de hardware, software o red.



¿Qué es Google Spanner?

Google Spanner es una base de datos distribuida globalmente que ofrece una gran consistencia y escalabilidad horizontal.

¿Cuales son las diferencias entre un Backup físico y lógico?

Físicos: copian directamente los archivos de datos del sistema.

Lógicos: exportan la información en formato legible (SQL, JSON...).

Tipo	Título / recurso	Autor(es) / entidad	
Paper de consenso (Raft)	*“In Search of an Understandable Consensus Algorithm”*	Ongaro & Ousterhout	Stanford University
Paper clásico (Paxos)	*“The Part-Time Parliament”*	Leslie Lamport	lamport.azurewebsites.net
Whitepaper Oracle / Data Guard	*Oracle Data Guard Technical White Paper*	Oracle	Oracle
Arquitecturas de alta disponibilidad Oracle	*Oracle MAA Reference Architectures*	Oracle	Oracle
Paper de comparación Paxos vs Raft	*Paxos vs Raft: Have we reached consensus on distributed consensus?*	Howard & Mortier	arXiv
Manual técnico Oracle RAC	*Oracle RAC Technical Architecture*	Oracle	Oracle
Comparativas algoritmo de consenso	*Paxos vs Raft*	varios autores	arXiv