INGENIERÍA EN SISTEMAS DE LA INFORMACIÓN

REDIS

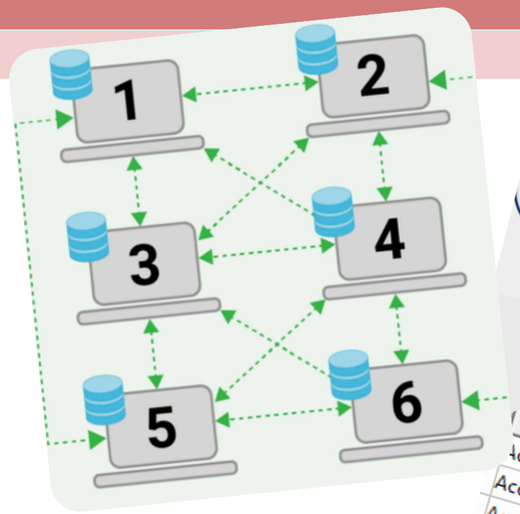
BASE DE DATOS I

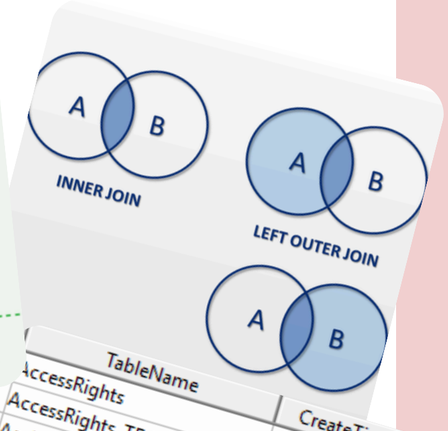
Miembros del equipo 4:

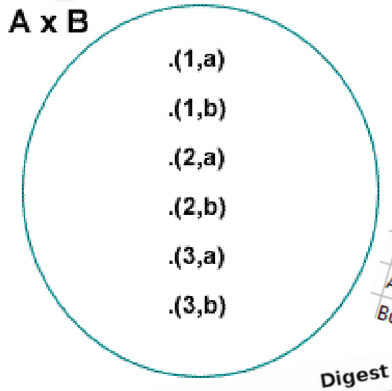
- Bustamante Cruz Erick
- López Corella David Antonio
- Villa Icedo Jesús Francisco

DESCRIPCIÓN

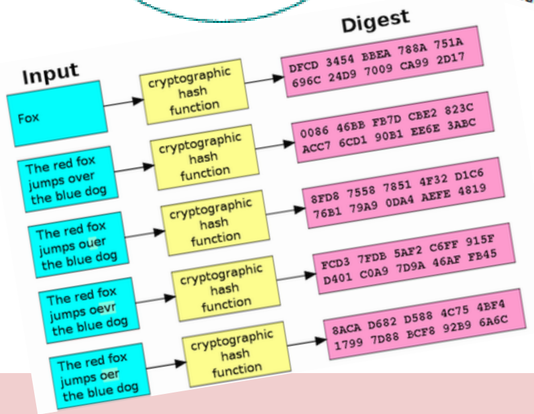
Es un sistema de gestión de base de datos NoSQL en memoria, que además funciona como sistema de caché y broker de mensajes. Ofrece varias estructuras de datos como cadenas, listas, conjuntos, hashes, conjuntos ordenados, bitmaps, y más.

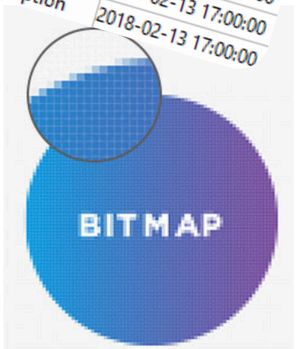




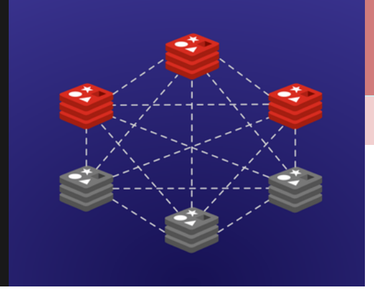


TableName	CreateTimeStamp
AccessRights	2018-02-13 17:00:00
AccessRights_TD14	2018-02-13 17:00:00
AccLogRuleTbl	2018-02-13 17:00:00
AccLogRuleTbl_TD14	2018-02-13 17:00:00
AccLogRuleTbl_TD15	2018-02-13 17:00:00
AccLogTbl	2018-02-13 17:00:00
AccLogTbl_TD14	2018-02-13 17:00:00
AccLogTbl_TD15	2018-02-13 17:00:00
Accounts	2018-02-13 17:00:00
Acctg	2018-02-13 17:00:00
ArchiveLoggingObjsTbl	2018-02-13 17:00:00
AsgdSecConstraints	2018-02-13 17:00:00
BusinessCalendarException	2018-02-13 17:00:00





REDIS SENTINEL



FUNCIONALIDADES

Sus funcionalidades principales incluyen almacenamiento en memoria para velocidad extrema, persistencia opcional en disco para durabilidad, replicación asíncrona, transacciones atómicas, soporte para scripting con Lua, mensajería pub/sub y alta disponibilidad mediante Redis Sentinel y Redis Cluster. Esto permite a los usuarios construir aplicaciones rápidas, escalables y confiables con acceso directo a datos mediante claves únicas.

Redis

HISTORIA BREVE

Redis fue creado en 2009 por Salvatore Sanfilippo para necesidades de real-time; la aplicación necesitaba mostrar en una interfaz de cada usuario el numero de visitas en tiempo real. Así nacio redis, que es un acrónimo de Remote Dictionary Server (Servidor de Diccionario Remoto). En esencia, era exactamente eso: un diccionario (o estructura de clave-valor) al que se podía acceder de forma remota a través de la red, manteniendo todos los datos en memoria para una velocidad insuperable.

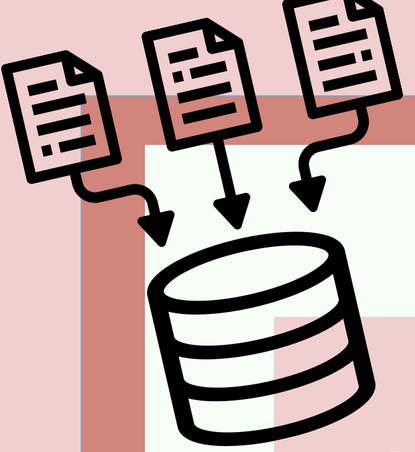
FUNCIONES DEL SMBD

1.- Almacenamiento, recuperación y actualización de datos.

- Mecanismos: Strings, lists, hash, sets, zsets.
- Algoritmos/procesos: skiplist y hash tables internas; lectura/escritura atómica por comando.

2.-Soporte de transacciones.

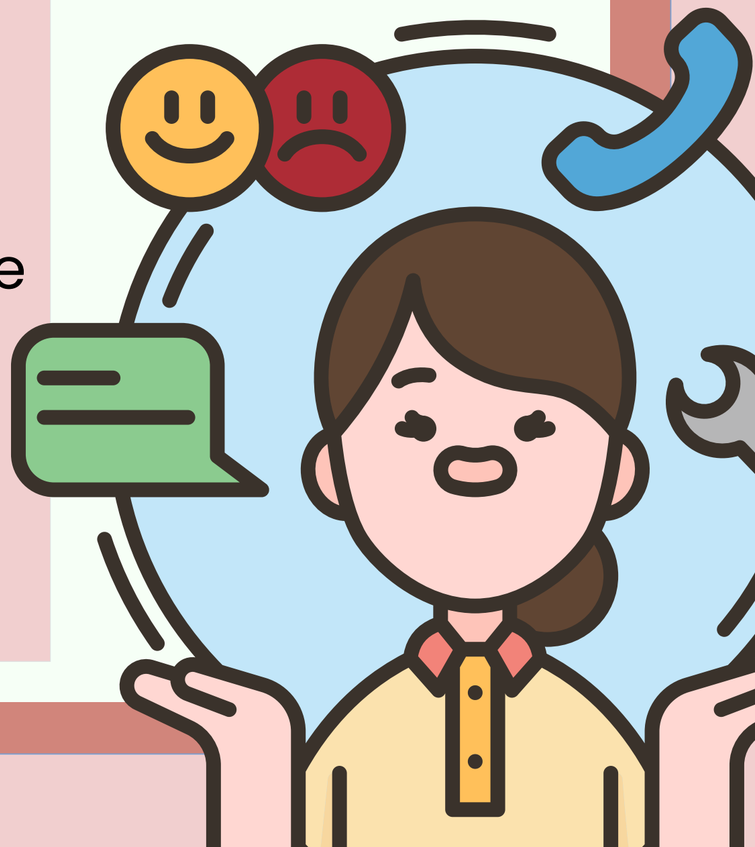
- MULTI/EXEC para ejecutar secuencias de comandos de forma aislada.
- Consistencia a nivel clave/secuencia y menor round-trip.
- También permite pipelines para enviar múltiples comandos sin esperar respuesta individualmente, mejorando rendimiento.



FUNCIONES DEL SMBD

3.- Servicios de recuperación.

- Redis puede restaurar datos mediante snapshots periódicos (RDB) o mediante el archivo append-only (AOF) que guarda cada operación modificadora recibida. Además, la replicación asíncrona permite tener réplicas que facilitan conmutación por error y alta disponibilidad, contribuyendo a la robustez del sistema y mínima pérdida de datos.



FUNCIONES DEL SMBD

4.-Servicios de control de concurrencia.

- Hilo único para el event loop, lo que serializa comandos y evita races dentro del servidor; WATCH habilita sincronización optimista; Lua corre de forma exclusiva.

5.-Servicios de autorización.

- En sus últimas versiones, Redis ha ampliado controles de acceso con listas ACL (Access Control Lists) que permiten definir permisos precisos y separar usuarios con diferentes niveles de acceso, por ejemplo, uno que solo pueda leer, otro que pueda escribir, y otro con acceso total.

MANEJO DE DDL

Para Redis el DDL (Lenguaje de definición de datos) es crear/eliminar estructuras o namespaces logicos, y el uso de diferentes tipos de datos clave-valor que Redis soporta: strings, hashes, listas, sets, sorted sets, bitmaps, hyperloglogs, streams, etc.

Ejemplos "análogo a DDL" en Redis:

- DEL (clave): elimina una clave y su estructura asociada.
- EXPIRE (clave tiempo): define el tiempo de vida de una clave, modificando su persistencia.
- RENAME clave_nueva clave_antigua: cambia el nombre de una clave.
- TYPE clave: permite conocer el tipo de estructura o dato que representa una clave.

MANEJO DE DML

Para DML(Lenguaje de manipulación de datos) en Redis, son operaciones de lectura/escritura sobre claves (set/get), hashes(hset/hget), listas, streams, etc.

Por ejemplo:

- SET user:1 "Ana" → guardar un valor (insertar).
- GET user:1 → leer un valor (consultar).
- HSET user:1 nombre "Ana" → insertar o actualizar un campo en un hash.
- HGET user:1 nombre → leer un campo de un hash.
- LPUSH lista:usuarios "Ana" → insertar un valor al inicio de una lista.

MANEJO DE DDL / DML

Importancia

La gestión de datos en Redis es fundamental para su rendimiento, ya que aunque carece de un lenguaje formal de definición de estructuras (DDL), el diseño implícito mediante la elección del tipo de dato correcto (strings, hashes, sets, etc.) determina la funcionalidad y eficiencia de las operaciones. Todas las manipulaciones de datos (DML) —inserciones, consultas o actualizaciones— se ejecutan en memoria con comandos atómicos de alta velocidad, lo que hace de Redis una herramienta clave para casos de uso que requieren respuesta inmediata, como caché, colas, sistemas de sesiones o contadores en tiempo real.

COMPATIBILIDAD DEL SISTEMA OPERATIVO

En Sistemas Populares

- **Linux:** Óptimo para producción. Requiere configurar *vm.overcommit_memory=1* y desactivar Transparent Huge Pages para evitar latencias.
- **BSD (Free/Open/Net):** Muy estable. Pocos requisitos específicos, funciona excelente por defecto.
- **macOS:** Ideal para desarrollo, **no para producción**. Limitación: Persistencia en disco (RDB/AOF) mucho más lenta que en Linux.
- **Windows:** Funcional. **Limitación:** Puede ir detrás en features y rendimiento a diferencia de Linux. Usar WSL2 o Docker para desarrollo.

COMPATIBILIDAD DEL SISTEMA OPERATIVO

En Sistemas de Nicho

- **Solaris/Illumos:** Funciona pero es de nicho. Ventaja: Soporta herramientas avanzadas como DTrace para profiling.
- **Embebidos/IoT (Linux/BSD):** Limitación principal: Los recursos (RAM y CPU) del hardware definen el tamaño máximo del dataset.
- **Mainframes (AIX, Linux on IBM Z):** Requisito: Compilación específica para la arquitectura (s390x, POWER). Funciona para cargas enterprise.
- **Otros (Haiku, MINIX):** Limitación: Puertos no oficiales, experimentales y sin soporte. No para producción.

SOPORTE DE LENGUAJES DE PROGRAMACIÓN

Redis soporta muchos lenguajes gracias a las múltiples bibliotecas cliente que realizan llamadas a comandos Redis, , simplificando la integración y desarrollo con esta base de datos en memoria.

1. JavaScript/Node.js: ioredis.
2. Python: redis-py.
3. Java: Jedis, Lettuce, Redisson..
4. NET/C#: StackExchange.Redis.
5. Go: go-redis.
6. PHP, Ruby, Rust: phpredis/Predis.

¿CÓMO INTERACTÚAN LOS LENGUAJES CON REDIS?

Usan bibliotecas cliente diseñadas para cada lenguaje que encapsulan la comunicación con el servidor Redis mediante comandos estándar (SET, GET, DEL, etc.).

Abstraen detalles de protocolo y conexión, facilitando una interacción sencilla y eficiente con la base de datos.

Ejemplo:

```
// Java con Jedis
Jedis jedis = new Jedis("localhost");
Transaction t = jedis.multi();
t.set("clave1", "valor1");
t.set("clave2", "valor2");
t.exec();
```

Java utiliza la biblioteca Jedis, donde se crean objetos Jedis para conectarse y enviar comandos.

STREAMING

LIVE

APLICACIONES EN EL MUNDO REAL

Redis se usa en aplicaciones que requieren de alta velocidad, disponibilidad y capacidad de manejar muchos usuarios al mismo tiempo.

Sus usos principales son:

- **Caché**: guardar datos en memoria para responder más rápido que con una base de datos tradicional.
- **Gestión de sesiones:** almacenar información de usuarios activos.
- **Colas de mensajes:** organizar tareas que deben procesarse en orden.
- **Rankings y conteos en tiempo real:** muy usado en juegos y sistemas de estadísticas.
- **Streaming:** enviar notificaciones o eventos en tiempo real entre servicios.

APLICACIONES EN EL MUNDO REAL

Empresas como X, Airbnb, Tinder, Yahoo, Adobe, Amazon, OpenAI han sido mencionadas como usuarias de Redis en infraestructura crítica y de alto rendimiento.

- **BioCatch:** utiliza Redis Enterprise en Azure para almacenar perfiles de comportamiento y datos críticos. Procesa 300 millones de claves y trabaja con 3 petabytes de datos, sin tiempos de inactividad.

Ejemplos de uso típicos: sesiones web, contadores, colas de trabajo, rate-limiters con Lua, pub/sub para mensajería y streams para eventing.

REFERENCIAS

- Redis Labs. (2025). *About Redis*. Redis. https://redis.io/about/
- IBM Corporation. (10 de mayo, 2024). *What is Redis explained? IBM Think*. IBM. https://www.ibm.com/think/topics/redis
- Mildenberger, T. (19 de diciembre, 2023). *Everything about Redis*. Contabo Blog. https://contabo.com/blog/everything-about-redis/
- IONOS. (30 de septiembre, 2020). *Tutorial de Redis | Primeros pasos con la base de datos*. IONOS. https://www.ionos.mx/digitalguide/hosting/cuestiones-tecnicas/redis-tutorial-paso-a-paso/
- Amazon Web Services. (s. f.). *¿Qué es Redis?*. AWS. https://aws.amazon.com/es/elasticache/what-is-redis/
- Fanchi, C. (9 de diciembre, 2022). *Redis: What it is, what it does, and why you should care*. Backendless. https://backendless.com/redis-what-it-is-what-it-does-and-why-you-should-care/
- Google Cloud. (s.f.). *Bibliotecas compatibles con Redis*. https://cloud.google.com/memorystore/docs/redis/libraries?hl=es-419