Apuntes de Redis – Ampliación de Bases de Datos

> **Objetivo**: Tener a mano toda la teoría, comandos y *patterns* que suelen caer en el examen, junto con una batería de ejercicios resueltos para practicar.

---

1. Introducción y arquitectura

- **Redis** (REmote DIctionary Server) es una base de datos clave-valor en memoria, extremadamente rápida, que también actúa como *cache* y sistema de mensajería (*pub/sub*). filecite turn2file1
- **Single-threaded** (comandos atómicos), pero escala horizontalmente vía **Cluster** y replicas **master-replica**. Persistencia opcional (RDB, AOF).
- Mediante `CONFIG SET` puedes ajustar políticas de expiración, memoria máxima y *eviction policy*. Ej.: `CONFIG SET maxmemory 150mb`.

---

2. Tipos de datos y comandos esenciales

2.1 Cadenas (Strings)

Acción	Comando	Ejemplo
Crear / sobrescribir	`SET clave valor [EX seg] [NX XX]`	`SET session:123 "{json}" EX 3600 NX`
Leer	`GET clave`	`GET session:123`
Incrementar	`INCRBY contador 5`	`INCRBY visita:home 1`
Expirar / TTL	`EXPIRE clave 60`	`EXPIRE temp:doc 60`

2.2 Listas (List)

Acción	Comando	Nota
Cola FIFO	`RPUSH cola val`	Añade al final
Pila LIFO	`LPUSH pila val`	Añade al inicio
Obtener rango	`LRANGE lista 0 -1`	Devuelve toda la lista
Bloqueo pop	`BLPOP lista 0`	Espera indefinido
Mover	`LMOVE src dst RIGHT LEFT`	Implementa listas circulares

2.3 Hashes (HSET/HGET)

```
HSET user:42 nombre "Ana" edad 30
HMGET user:42 nombre edad     # → Ana 30
HINCRBY user:42 puntos 10
```

2.4 Sets (SADD/SMEMBERS)

```
SADD tags:post:9 redis nosql cache
SISMEMBER tags:post:9 cache      # → 1
SDIFFSTORE soloRedis tags:redis tags:nosql
```

2.5 Sorted Sets ⭐(muy examinados)

```
# Lista de espera pacientes (score = prioridad)
ZADD lista_espera 0 Pedro 23.2 Juan 19.5 Lola 13 Julia  # examen

# Rango por score (10 incluido, 20 excluido)
ZRANGEBYSCORE lista_espera 10 (20                       # examen

# Paciente con mayor prioridad (score más bajo) y lo elimina
BZPOPMIN lista_espera 0                                 # examen
```

Otros imprescindibles: `ZSCORE`, `ZRANK`, `ZCOUNT`, `ZREMRANGEBYSCORE`, `ZUNIONSTORE`, `ZINTERSTORE`.

2.6 Bitmaps, HyperLogLog, Streams (mención)

- `SETBIT`, `BITCOUNT` para flags compactos.
- `PFADD` / `PFCOUNT` para aproximar cardinalidad enorme.
- `XADD` / `XREADGROUP` para colas *streaming*.

---

3. Atomicidad y concurrencia

- **Pipelining**: agrupa comandos y recibe respuestas luego. Acelera 3-10× en WAN.
- **Transacciones** con `MULTI … EXEC` (todo-o-nada). Añade `WATCH key1 key2` para control *optimistic locking*.
- **Lua scripting** (`EVAL`) para operaciones complejas atómicas.

---

4. Pub/Sub

```
# Terminal 1 – suscriptor universal
PSUBSCRIBE canal.*
```

```
# Terminal 2 – publicador
PUBLISH canal.chat "Hola 👋"
```

Útil para invalidar cache o eventos en tiempo real. filecite turn2file1

---

5. Configuración de Cache y memoria

Parámetro	Ejemplo	Comentario
Límite RAM	`CONFIG SET maxmemory 150mb`	150 MB para cache práctica LocalStore filecite turn2file0
Política Eviction	`CONFIG SET maxmemory-policy allkeys-lru`	Expulsa la clave menos usada
Tiempo vida por defecto	`SET key val EX 86400`	24 h cache

Tip examen: Se suele pedir «noeviction frente a allkeys-lru» y cómo establecer expiración.

---

6. Patrón Cola de tareas (práctica empaquetado)

```
# Productor: encolar idCompra
LPUSH empaquetar 99

# Trabajador principal (wait forever)
BLPOP empaquetar 0   # devuelve [key, 99]

# Trabajador secundario (timeout 60 s)
BLPOP empaquetar 60
```

Cuando un trabajador procesa, lanza otro con `id = id_creador+1`. Se controla con scripts de **hilos** en Python usando `threading` + `redis-py`.

---

7. Ejercicios de preparación para el examen (con soluciones)

#	Enunciado	Solución	Explicación breve
1	Añade a un *sorted set* de ranking los usuarios `u1-u5` con puntuación igual a su nivel (10,20,30,40,50).	`ZADD ranking 10 u1 20 u2 30 u3 40 u4 50 u5`	Orden ascendente por score.
2	Suma 5 puntos al usuario `u3` y devuelve su nueva posición.	`ZINCRBY ranking 5 u3` seguido de `ZRANK ranking u3`	`ZINCRBY` actualiza score atómicamente.

#	Enunciado	Solución	Explicación breve
3	Devuelve los 3 mejores usuarios con score y sin score.	`ZREVRANGE ranking 0 2 WITHSCORES`	REVRANGE = mayor → menor.
4	Cola prioritaria «tickets»: inserta (score = unixTime) y extrae más antiguo.	`ZADD tickets 1625000000 t1` + `ZPOPMIN tickets`	UnixTime crece ⇒ score min = más antiguo.
5	Usar `SCAN` para listar todas las claves `session:*` de 20 en 20.	`SCAN 0 MATCH session:* COUNT 20`	Cursor incremental, evita *block*.
6	Implementa contador global de visitas que nunca baje de 0.	`DECRBY visita 1` & check `GET visita`, o usa script Lua para condicional.	
7	Calcula cuántos visitantes únicos tuvo tu web hoy (aprox.).	`PFADD uniq:2025-06-21 ip1 ip2 …` luego `PFCOUNT uniq:2025-06-21`	HyperLogLog filecite turn2file1
8	Mueve pedido de `pendientes` a `procesados` de manera atómica.	`LMOVE pendientes procesados RIGHT LEFT`	Elimina de una lista y mete en otra sin *race*.
9	Elimina del set `inactivos` los usuarios que hayan iniciado sesión (appear en `activos`).	`SDIFFSTORE inactivos inactivos activos`	Guarda diff inplace.
10	Expirar automáticamente las claves `temp:*` en 1 h al crearlas.	Prefijo con `SET temp:123 val EX 3600` o usa `redis-expire` script.	
11	Configura Redis para: a) 256 MB máx.; b) expulsión por TTL próximo.	`CONFIG SET maxmemory 256mb` + `CONFIG SET maxmemory-policy volatile-ttl`	Eviction sólo en claves con TTL.
12	Devuelve 2 pacientes de prioridad 0-15 (incluida) orden descendente de urgencia.	`ZPOPMIN lista_espera 2`	Si score = urgencia invertida.

Ejercicio práctico completo – Gestión lista de espera hospital

1. **Admitir paciente**

```
ZADD lista_espera <prioridad> <nombre>
```

1. **Visualizar top-N urgentes**

```
ZPOPMIN lista_espera N   # o ZRANGE lista_espera 0 N-1 WITHSCORES
```

1. **Rebajar prioridad (empeora)**

```
ZINCRBY lista_espera 5 Juan   # +5 = menos urgente si score bajo = más prioritario
```

1. **Pacientes prioridad 10 ≤ p < 20**

```
ZRANGEBYSCORE lista_espera 10 (20
```

Preguntas flash de teoría

1. **¿Qué comando devuelve y elimina el elemento con score más alto?** → `ZPOPMAX`.
2. **¿Qué modos de persistencia ofrece Redis?** → RDB, AOF y mezcla (fsync append-only + snapshots).
3. **¿Para qué sirve** `` `**?** → Implementar transacciones optimistas; aborta `` EXEC` si cambia alguna clave vigilada.
4. **¿Política allkeys-lru afecta a claves con TTL?** → Sí, todas las claves se consideran candidatas.
5. **¿** `** con opciones **` **garantiza qué?** → Crea la clave solo si no existe y caduca a los 1000 ms.

---

8. Buenas prácticas

- Prefijo de claves: `tipo:subtipo:id` → facilita SCAN/MATCH.
- Mantén TTL coherente; no dejes basura en memoria.
- Usa `PIPELINE` o `MULTI` para *round-trip* largos.
- Evita **big keys** (>10 MB) – bloquean hilo único.
- Mide con `INFO memory` y `MEMORY USAGE key`.

¡Ya tienes un compendio completo para clavar el examen de Redis! Practica cada comando en tu `redis-cli` y adapta los patrones a los enunciados concretos que te planteen.

Apuntes de Redis Ampliación de Bases de Datos
[contenido original mantenido]

9. Ejercicios adicionales estilo examen (ampliación)
9.1 Sorted Sets (ZADD, ZRANGE, ZINCRBY, ZPOPMIN, ZREMRANGEBYSCORE)

```
# Crear ranking de usuarios
ZADD ranking 150 Ana 200 Luis 180 Marta

# Incrementar puntuación
ZINCRBY ranking 25 Ana

# Mostrar usuarios con puntuación entre 160 y 210
ZRANGEBYSCORE ranking 160 210

# Quitar a los que tengan menos de 160 puntos
ZREMRANGEBYSCORE ranking -inf 159

# Sacar al más prioritario (menor puntuación)
ZPOPMIN ranking
```

9.2 Listas (RPUSH, BLPOP, LMOVE)

```
# Encolar trabajos
RPUSH trabajos t1 t2 t3

# Consumir el primero con espera infinita
BLPOP trabajos 0

# Mover de 'pendientes' a 'procesados'
LMOVE pendientes procesados RIGHT LEFT
```

9.3 Hashes y Sets

```
# Guardar perfil de usuario
HSET usuario:42 nombre "Pedro" edad 34 email "pedro@example.com"

# Verificar si un usuario está en la lista de acceso
SISMEMBER acceso_activo usuario:42

# Añadir usuarios activos
SADD acceso_activo usuario:42 usuario:43

# Quitar inactivos
SDIFFSTORE solo_activos acceso_activo acceso_inactivo
```

9.4 Expiraciones y HyperLogLog

```
# Establecer TTL de 1 hora
SET temp:clave123 "valor" EX 3600

# Aproximar visitantes únicos
```

```
PFADD visitas:2025-06-21 ip1 ip2 ip3
PFCOUNT visitas:2025-06-21
```

9.5 Pub/Sub

```
# Suscripción a canal
PSUBSCRIBE notificaciones.*

# Enviar mensaje
PUBLISH notificaciones.alerta "Servidor caído"
```

9.6 CONFIG SET y SCAN

```
# Limitar uso de memoria a 64MB
CONFIG SET maxmemory 67108864

# Política de expulsión
CONFIG SET maxmemory-policy allkeys-lru

# Buscar claves por patrón
SCAN 0 MATCH temp:* COUNT 100
```

9.7 Transacciones con WATCH + MULTI/EXEC

```
# Asegurar stock positivo al descontar
WATCH stock:prod123
GET stock:prod123
# Si es >0, proceder
MULTI
DECR stock:prod123
EXEC
# Si otro cliente lo cambia, EXEC no se ejecuta
```

9.8 Mixtos con patrones

```
# Cola de emails
LPUSH emails email_1
BLPOP emails 0

# Clasificación por urgencia en ZSET
ZADD urgencias 0 pacienteA 3 pacienteB 5 pacienteC
ZPOPMIN urgencias 2

# Almacenar eventos
XADD eventos * tipo=login user=Ana
```

Estos ejemplos cubren los tipos de ejercicios que pueden aparecer tanto en la parte teórica como práctica del examen, incluyendo control de concurrencia, configuraciones, estructuras complejas y comandos menos habituales.