Redis Riferimento Rapido

Stringhe, liste, set, hash, pub/sub, persistenza

Connessione

CLI

```
redis-cli
redis-cli -h 127.0.0.1 -p 6379
redis-cli -a password -n 2
redis-cli --tls -u rediss://user:pass@host:6380
```

Connessione con Driver (Python)

```
import redis
r = redis.Redis(host='localhost', port=6379, db=0)
r.set('key', 'value')
print(r.get('key'))
```

Informazioni Server

```
PING           -- restituisce PONG
INFO server    -- dettagli server
INFO memory    -- utilizzo memoria
DBSIZE         -- numero di chiavi nel db corrente
```

Stringhe

Operazioni di Base

```
SET name "Alice"
GET name
SET counter 100
MSET a 1 b 2 c 3
MGET a b c
```

Operazioni Numeriche

```
INCR counter        -- 101
INCRBY counter 10   -- 111
DECR counter        -- 110
DECRBY counter 5    -- 105
INCRBYFLOAT price 2.5
```

Comandi sulle Stringhe

`SET key val`	Imposta valore stringa
`GET key`	Ottieni valore stringa
`SETNX key val`	Imposta solo se la chiave non esiste
`SETEX key sec val`	Imposta con scadenza in secondi
`APPEND key val`	Aggiunge al valore esistente
`STRLEN key`	Lunghezza del valore stringa

Liste

Operazioni sulle Liste

```
LPUSH queue "first"
RPUSH queue "last"
LRANGE queue 0 -1    -- tutti gli elementi
LPOP queue
RPOP queue
```

Comandi sulle Liste

`LPUSH / RPUSH`	Inserisce a sinistra / destra della lista
`LPOP / RPOP`	Estrae da sinistra / destra
`LRANGE key start stop`	Ottieni intervallo di elementi
`LLEN key`	Lunghezza della lista
`LINDEX key idx`	Elemento all'indice
`LREM key count val`	Rimuovi count occorrenze di val
`BLPOP key timeout`	Pop bloccante (per code)

Set e Set Ordinati

Operazioni sui Set

```
SADD tags "python" "redis" "docker"
SMEMBERS tags
SISMEMBER tags "python"   -- 1 (vero)
SREM tags "docker"
SCARD tags                -- conteggio
```

Operazioni sui Set

```
SUNION set1 set2       -- unione
SINTER set1 set2       -- intersezione
SDIFF set1 set2        -- differenza
```

Operazioni sui Set Ordinati

```
ZADD leaderboard 100 "Alice" 85 "Bob"
ZRANGE leaderboard 0 -1 WITHSCORES
ZREVRANGE leaderboard 0 2
ZSCORE leaderboard "Alice"
ZRANK leaderboard "Alice"    -- rank in base 0
```

Comandi sui Set Ordinati

`ZADD key score member`	Aggiunge membro con punteggio
`ZRANGE key start stop`	Intervallo per rank (dal basso)
`ZREVRANGE key start stop`	Intervallo per rank (dall'alto)
`ZINCRBY key incr member`	Incrementa punteggio membro
`ZRANGEBYSCORE key min max`	Intervallo per valore del punteggio
`ZCARD key`	Numero di membri

Hash

Operazioni sugli Hash

```
HSET user:1 name "Alice" age 30
HGET user:1 name
HGETALL user:1
HMSET user:2 name "Bob" age 25
HMGET user:1 name age
```

Comandi sugli Hash

`HSET key field val`	Imposta campo hash
`HGET key field`	Ottieni campo hash
`HGETALL key`	Ottieni tutti i campi e valori
`HDEL key field`	Elimina campo hash
`HEXISTS key field`	Verifica esistenza campo
`HINCRBY key field n`	Incrementa valore campo
`HKEYS key`	Tutti i nomi dei campi
`HLEN key`	Numero di campi

Chiavi e Scadenza

Comandi sulle Chiavi

`KEYS pattern`	Trova chiavi corrispondenti al pattern (lento)
`SCAN cursor MATCH pat`	Itera le chiavi in modo incrementale (sicuro)
`EXISTS key`	Verifica se la chiave esiste
`DEL key`	Elimina chiave
`TYPE key`	Tipo di dato della chiave
`RENAME key newkey`	Rinomina una chiave

Comandi di Scadenza

```
EXPIRE key 3600       -- scade in 1 ora
PEXPIRE key 5000      -- scade in 5000 ms
TTL key               -- secondi alla scadenza
PTTL key              -- ms alla scadenza
PERSIST key           -- rimuovi scadenza
```

Pattern di Chiave

```
SET session:abc123 "data" EX 1800
-- EX = secondi, PX = millisecondi
-- NX = solo se non esiste
-- XX = solo se esiste
SET lock:order42 "owner" NX EX 10
```

Pub/Sub

Pub/Sub di Base

```
-- Subscriber (terminale 1)
SUBSCRIBE news alerts

-- Publisher (terminale 2)
PUBLISH news "Breaking: Redis 8 released"
```

Sottoscrizione per Pattern

```
PSUBSCRIBE news.*
-- corrisponde a news.tech, news.sports, ecc.
```

Comandi Pub/Sub

`SUBSCRIBE channel`	Ascolta messaggi sul canale
`PUBLISH channel msg`	Invia messaggio al canale
`PSUBSCRIBE pattern`	Sottoscrive a un pattern
`UNSUBSCRIBE channel`	Smetti di ascoltare
`PUBSUB CHANNELS`	Elenca i canali attivi

Transazioni

MULTI / EXEC

```
MULTI
SET balance:1 900
SET balance:2 1100
EXEC        -- esegue atomicamente
```

Lock Ottimistico

```
WATCH balance:1
val = GET balance:1   -- legge valore corrente
MULTI
SET balance:1 (val - 100)
EXEC
-- EXEC restituisce nil se balance:1 è cambiato
```

Comandi di Transazione

`MULTI`	Inizia il blocco transazione
`EXEC`	Esegue i comandi in coda
`DISCARD`	Scarta i comandi in coda
`WATCH key`	Monitora la chiave per cambiamenti (lock ottimistico)
`UNWATCH`	Dimentica tutte le chiavi monitorate

Persistenza

Snapshot RDB

```
SAVE            -- snapshot sincrono
BGSAVE          -- snapshot in background
LASTSAVE        -- timestamp dell'ultimo salvataggio
```

AOF (Append Only File)

`appendonly yes`	Abilita AOF in redis.conf
`appendfsync always`	Fsync ad ogni scrittura (più sicuro, più lento)
`appendfsync everysec`	Fsync ogni secondo (raccomandato)
`appendfsync no`	Lascia decidere all'OS (più veloce, più rischioso)

Redis Riferimento Rapido

Comandi di Persistenza

```
CONFIG GET save
CONFIG SET save "900 1 300 10"
-- snapshot se 1 modifica in 900s o 10 in 300s
BGREWRITEAOF    -- riscrive AOF in background
```

Pattern Comuni

Lock Distribuito

```
SET lock:resource "owner-id" NX EX 30
-- NX = acquisisci solo se non in possesso
-- EX 30 = rilascio automatico dopo 30s
DEL lock:resource  -- rilascio esplicito
```

Rate Limiter

```
key = "rate:user:42"
INCR key
EXPIRE key 60   -- finestra di 60 secondi
-- rifiuta se GET key > max_requests
```

Pattern di Cache

```
val = GET "cache:user:1"
if val is nil:
  val = fetch_from_db(1)
  SET "cache:user:1" val EX 300
```

Storage di Sessione

```
HSET sess:abc uid 42 role "admin"
EXPIRE sess:abc 1800   -- TTL 30 min
HGETALL sess:abc
```