Redis + Python

Thursday, February 06, 2025 9:29 AM

Redis Command List

- Full List > here <
- Use Filter to get to command for the particular data structure you're targeting (list, hash, set, etc.)
- Redis.py Documentation > here <
- The next slides are not meant to be an exhaustive list of commands, only some highlights. Check the documentation for a complete list.

String Commands

```
# r represents the Redis client object
r.set('clickCount:/abc', 0)
val = r.get('clickCount:/abc')
r.incr('clickCount:/abc')
ret_val = r.get('clickCount:/abc')
print(f'click count = {ret_val}')

# r represents the Redis client object
redis_client.mset({'key1': 'val1',
                   'key2': 'val2',
                   'key3': 'val3'})
print(redis_client.mget('key1',
                        'key2',
                        'key3'))
# returns as list ['val1', 'val2', 'val3']
```

- set(), mset(), setex(), msetnx(), setnx()
- get(), mget(), getex(), getdel()
- incr(), decr(), incrby(), derby()
- strlen(), append()

List Commands

```
# create list: key = 'names'
# values = ['mark', 'sam', 'nick']
redis_client.rpush('names',                 # Right push
                   'mark', 'sam', 'nick')


# prints ['mark', 'sam', 'nick']
print(redis_client.lrange('names', 0, -1))
```

- lpush(), lpop(), lset(), lrem()
- rpush(), rpop()
- lrange(), llen(), lpos()
- Other commands include moving elements between lists, popping from multiple lists at the same time, etc.

Hash Commands

```
redis_client.hset('user-session:123',
     mapping={'first': 'Sam',
              'last': 'Uelle',
              'company': 'Redis',
              'age': 30
})


# prints:
#{'name': 'Sam', 'surname': 'Uelle', 'company': 'Redis', 'age': '30'}
print(redis_client.hgetall('user-session:123'))
```

- hset(), hget(), hgetall()
- hkeys()
- hdel(), hexists(), hlen(), hstrlen()

Redis Pipelines

- Helps avoid multiple related calls to the server → less network overhead

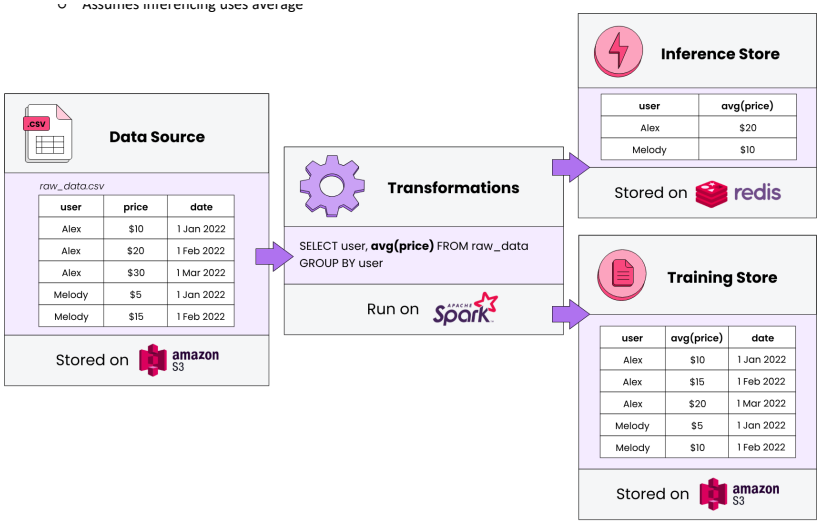
```
r = redis.Redis(decode_responses=True)
pipe = r.pipeline()

for i in range(5):
   pipe.set(f"seat:{i}", f"#{i}")

set_5_result = pipe.execute()
print(set_5_result)  # >>> [True, True, True, True, True]

pipe = r.pipeline()

# "Chain" pipeline commands together.
get_3_result= pipe.get("seat:0").get("seat:3").get("seat:4").execute()
print(get_3_result)  # >>> ['#0', '#3', '#4']
```

Redis in Context

- Why use redis instead of a MySQL?
  - Latency issues – accessing a value with a particular key in Redis is way faster than running a select statement on disk (rather than RAM)

- **Redis in ML (simplified)**
  - Assumes inferencing uses average

- Assumes inferencing uses average



- **Redis in DS/ML**