CSC582 - Data Warehouse and Mining Systems

Relational to Key-Value Database Migration

E-Commerce Database with Redis Cluster

Component	Description
Database	E-Commerce
Target	Redis Cluster
Nodes	8 (4 Masters + 4 Replicas)
Partitioning	Hash-Based (CRC16)

December 2024

1. Introduction

This project demonstrates migrating a relational database to Redis, a key-value store. It covers schema design, mapping strategy, Redis Cluster configuration with sharding and replication, data migration, and query operations.

Objectives:

- Design E-Commerce relational database (3 tables)
- Map relational model to key-value structures
- Configure 8-node Redis Cluster
- Implement hash-based partitioning
- Set up replication for high availability
- Demonstrate key-value operations

2. Relational Model and Schema

2.1 Database Tables

CUSTOMERS Table:

Column	Type	Constraint
customer_id	INT	PRIMARY KEY
first_name	VARCHAR(50)	NOT NULL
last_name	VARCHAR(50)	NOT NULL
email	VARCHAR(100)	UNIQUE
city	VARCHAR(50)	

PRODUCTS Table:

Column	Type	Constraint
product_id	INT	PRIMARY KEY
product_name	VARCHAR(100)	NOT NULL
category	VARCHAR(50)	
price	DECIMAL(10,2)	NOT NULL
stock_quantity	INT	DEFAULT 0

ORDERS Table:

Column	Type	Constraint
order_id	INT	PRIMARY KEY
customer_id	INT	FOREIGN KEY
product_id	INT	FOREIGN KEY
quantity	INT	NOT NULL
total_amount	DECIMAL(10,2)	
status	VARCHAR(20)	

3. Mapping Relational to Key-Value

3.1 Mapping Strategy

- **Hashes** - Store table rows (customer:1, product:101)
- **Sets** - Secondary indexes (customers:by_city:riyadh)
- **Sorted Sets** - Range queries (orders:by_amount)

3.2 Key Naming Convention

Pattern	Example	Type
customer:{id}	customer:1	HASH
product:{id}	product:101	HASH
customers:by_city:{city}	customers:by_city:riyadh	SET
orders:by_customer:{id}	orders:by_customer:1	SET

4. Redis Cluster Configuration

4.1 Hash-Based Partitioning

Redis Cluster uses CRC16 hash function with 16,384 hash slots:

```
Algorithm: slot = CRC16(key) mod 16384

Example: customer:1 -> slot 15495 -> Master 4
```

4.2 Sharding (4 Masters)

Node	Port	Slots
Master 1	7001	0-4095
Master 2	7002	4096-8191
Master 3	7003	8192-12287
Master 4	7004	12288-16383

4.3 Replication (4 Replicas)

Master	Port	Replica	Port
Node 1	7001	Node 5	7005
Node 2	7002	Node 6	7006
Node 3	7003	Node 7	7007
Node 4	7004	Node 8	7008

5. Key-Value Operations

This code demonstrates how we retrieve values from Redis using our Key-Value schema.

```
def demo_key_value_queries():
    r = get_redis_client()

    # KV: Get customer 1 name and email
    name = get_attr(r, "customer", 1, "name")
    email = get_attr(r, "customer", 1, "email")
    print("customer 1:", name, email)

    # KV: Get product 101 name and price
    pname = get_attr(r, "product", 101, "name")
    price = get_attr(r, "product", 101, "price")
    print("product 101:", pname, price)

    # KV: Simulate JOIN for order 1001
    cid = int(get_attr(r, "order", 1001, "customer_id"))
    pid = int(get_attr(r, "order", 1001, "product_id"))
    qty = int(get_attr(r, "order", 1001, "quantity"))
    odate = get_attr(r, "order", 1001, "order_date")

    cname = get_attr(r, "customer", cid, "name")
    pname = get_attr(r, "product", pid, "name")
    price = float(get_attr(r, "product", pid, "price"))

    print(
        f"order 1001: customer={cname}, product={pname}, "
        f"date={odate}, qty={qty}, total={qty * price}"
    )

    # KV: List all orders from orders:all
    order_ids = r.smembers("orders:all")
    for oid in sorted(int(x) for x in order_ids):
        cid = int(get_attr(r, "order", oid, "customer_id"))
        cname = get_attr(r, "customer", cid, "name")
        print(f"order {oid} belongs to {cname}")
```

Sample Output:

```
customer 1: Ahmed Al-Rashid ahmed.rashid@email.com
product 101: Laptop Pro X1 4500.00

order 1001: customer=Ahmed Al-Rashid, product=Laptop Pro X1,
            date=2024-06-01, qty=1, total=4500.00

order 1001 belongs to Ahmed Al-Rashid
order 1002 belongs to Ahmed Al-Rashid
order 1003 belongs to Fatima Hassan
order 1004 belongs to Mohammed Ali
order 1005 belongs to Sara Omar
```

Helper Function:

```python
def get_attr(r, entity, id, attribute):
    """Retrieve a single attribute from a Redis hash."""
    key = f"{entity}:{id}"
    return r.hget(key, attribute)

def get_redis_client():
    """Connect to Redis cluster or single instance."""
    return redis.Redis(host='127.0.0.1', port=6379, decode_responses=True)
```

6. Data Migration

The migration process transfers data from relational to Redis:

```
def migrate_customers(r, customers):
    for c in customers:
        key = f"customer:{c['customer_id']}"
        r.hset(key, mapping={
            'name': f"{c['first_name']} {c['last_name']}",
            'email': c['email'],
            'city': c['city']
        })
        # Add to city index
        r.sadd(f"customers:by_city:{c['city'].lower()}", c['customer_id'])
        # Add to all customers set
        r.sadd("customers:all", c['customer_id'])

def migrate_orders(r, orders):
    for o in orders:
        key = f"order:{o['order_id']}"
        r.hset(key, mapping={
            'customer_id': o['customer_id'],
            'product_id': o['product_id'],
            'quantity': o['quantity'],
            'order_date': o['order_date'],
            'total_amount': o['total_amount']
        })
        # Add to orders:all set
        r.sadd("orders:all", o['order_id'])
        # Add to customer's orders index
        r.sadd(f"orders:by_customer:{o['customer_id']}", o['order_id'])
```

Migration Summary:

Table	Records
Customers	5
Products	6
Orders	8

7. Conclusion

This project successfully demonstrated:

- Designed 3-table E-Commerce relational database
- Mapped to Redis using Hashes, Sets, Sorted Sets
- Configured 8-node Redis Cluster (4+4)
- Implemented CRC16 hash-based sharding
- Set up automatic failover replication
- Migrated 19 records with referential integrity
- Demonstrated key-value queries simulating SQL JOINs

Project Achievements:

```
+------------------------------------------------------------------+
|                    PROJECT SUMMARY                               |
+------------------------------------------------------------------+
| Tables Designed:        3 (Customers, Products, Orders)          |
| Cluster Nodes:          8 (4 Masters + 4 Replicas)               |
| Hash Slots:             16,384 distributed across 4 masters      |
| Records Migrated:       19 total (5 + 6 + 8)                     |
| Operations Demonstrated: HGET, HSET, SADD, SMEMBERS, JOIN sim    |
+------------------------------------------------------------------+
```