HW3 Part 1: Relational vs Key-Value vs Document Design

Worked by Xiang Zhu & Boyuan Li

Queries:

1. Relational DB – One big table:

```
SELECT *
FROM products
WHERE
        (diameter   = "44mm"
        AND brand  = "Tommy Hilfiger"
        AND dial_color = "Beige");
```

2. Relational DB - Category tables:

```
SELECT *
FROM watches w
LEFT JOIN product p
ON w.product_id = p.product_id
WHERE
        (diameter = "44mm"
        AND brand = "Tommy Hilfiger"
        AND dial_color = "Beige");
```

3. Relational DB - Property table:

```
SELECT *
FROM
        (SELECT product_id
        FROM
                (SELECT product_id
                FROM category c JOIN property p1 ON p1.category_id = c.category_id
                WHERE
                        (c.category_name = "watches"
                        AND c.key = "diameter" AND c.value = "44mm") c1
```

```
        JOIN property p2 ON c1.product_id = p2.product_id
        WHERE p2.key = "brand" AND p2.value = "Tommy Hilfiger") c2
JOIN property p3 ON c2.product_id = p3.product_id
WHERE p3.key = "dial_color" AND p3.value = "Beige";
```

4. A Key-Value Store:

We store product information in a key-value store such as Redis.

Example:

```
ZADD category 'watch' 'hashmap1';
HSET hashmap1 diameter '44mm' brand 'Tommy Hilfiger' dial_color 'Beige';

ZADD category 'watch' 'hashmap2';
HSET hashmap1 diameter '22mm' brand 'Tommy Hilfiger' dial_color 'Blue';
```

We use ZADD with the “category” as the set name, the “product name” as the key , and the set name of HEST is the value of the ZADD.

5. Document Store using MongoDB:

```
db.products.find({"category": "watch",
        "diameter": "44mm",
        "brand": "Tommy Hilfiger",
        "dial_color": "Beige"})
```

Synopsis:

One big table is a good model to deal with the massive amounts of information that the company regularly deals in. We populate relevant columns with the properties of the product, leaving the irrelevant columns empty or NULL. If we do not need too many columns, this model could be fine. However, if we use One Big Table to store the product information for Amazon, we need so many columns and this can lead to the low performance of the system. If we need add new columns, we also need to add values for the previous product infromation. Category tables increase the complexity of the model. Because it need to join operation when query for a specific product, the operation would be pretty expensive. However, it provides a better data organization and consistence. Property table make the model more complex, but it allows user to find a specified product by creating nested queries and add special properties for

a product, such as MongoDB.

Key-Value model runs so fast because Redis is memory-based. The model is complex, but the queries are simple. As a key-value store, you will have a simple and limited set of operations with the database. But, in terms of performance, you should evaluate too how you will scale it and the ratio of read and writes. Depending the case, the key-value will be faster. In the Document Store Mode, we use MongoDB to store the product information as a single document. The data are schemaless. Even if the data model is complex, the queries are simple and efficient. Each document is isolated and does not need to join any other data. MongoDB lacks some features of relational database like transaction or join, but it gain the ability to scale horizontal easily and flexible schema that easy to manipulate with JSON like data format.