DS 4300: Large-Scale Storage and Retrieval
Product Catalogs with Relational and NoSQL
Prof. Rachlin

DESCRIPTION

This homework has two parts. In the first part of the assignment, we'll consider competing relational and document database designs for a large product catalog with thousands of product categories and tens of millions of products. You'll evaluate the design tradeoffs. In the second part of the assignment, you'll write an API search method to query a small product catalog that you create for test purposes.

Part 1: Relational vs Key-Value vs Document Design

Let's consider five options for managing a product catalog:

1. Relational DB – One big table: One row per product. We populate relevant columns with the properties of the product, leaving the irrelevant columns empty or NULL.

2. Relational DB - Category tables: There are two common tables plus one additional table for each category. The CATEGORY table is just a table of category names and ids. The PRODUCT table contains information common to all products like "Price" and "Availability". There is one row per product. In addition, we have one table for EACH product category, e.g., WATCHES, WINE, etc. that contains a finite set of attributes appropriate to each attribute

    - CATEGORY (category_id, category_name)
    - PRODUCT (product_id, price, is_available)
    - WATCHES (product_id, diameter, band_type, …)
    - WINE (product_id, winery, type, year_bottled, …)

3. Relational DB - Property table: We replace the category-specific tables (e.g., WATCHES) with a single common table: PROPERTY (property_id, product_id, category_id, key, value). For example, the key might be: "diameter" and the value could be "44 mm"

4. A Key-Value Store: We store products in a key-value store such as Redis. For example, each key might be a product code pointing to a hashmap structure. But how will you implement a search API?

5. Document Store using MongoDB: The item is rendered as a single document in MongoDB.

Now consider searching the WATCH category for all watches with ALL of the following three properties:

a) Diameter: 44mm
b) Brand: Tommy Hilfiger
c) Dial color: Beige

For each of the five models, give a SQL query, Redis command script (4th option – a general outline is fine), or a MongoDB query (5th option). For Redis explain your data model – what are the keys and what are the values? Write a couple of paragraphs comparing the different approaches with respect to the complexity of the data model, the query or application complexity, and inherent model limitations if any.

Part 2: A search API

This is intended as a small programming assignment just to get familiar with application programming using Mongo. I personally find PyMongo (a Mongo driver for Python) pretty convenient but you may implement your search API method in any language you wish using any client you wish. Your API function can be written with any signature you want, but a programmer should at *minimum* be able to specify a product category name, and a list of key/value pairs as filter criteria, sufficient to run the watch search in part 1. Your API should abstract out the fact that the database storage engine is MongoDB, just as we did with Assignments #1 and #2. For test data, it is fine to create your own small mongo database manually containing a handful of products and attributes.

SUBMISSION:

1. Queries and Synopsis for Part1 (I expect this to be at most one page.)
2. Code for part 2
3. Text output demonstration for part 2