Alexandria University

Faculty of Engineering

Shopping Research Porject Proposal

Supervision:

Dr. Bassem Mahmoud Mokhtar - EED

Researcher:

Abanoub Milad Nassief - CSED

Abstract

Developing an intelligent shopping system with efficient database management and intelligent various search techniques on the stored data.

System Features

- Shopping by categories, subcategories, items, promotions and stores.
- Custom item search
- Smart search based on machine learning by entering full text describing item features or specifications. System recommends the most fitting items.
- Register for items and promotinos and getting notified when they become available

Research and development phases

1. Requirements gathering and analysis (Requirement Specification)

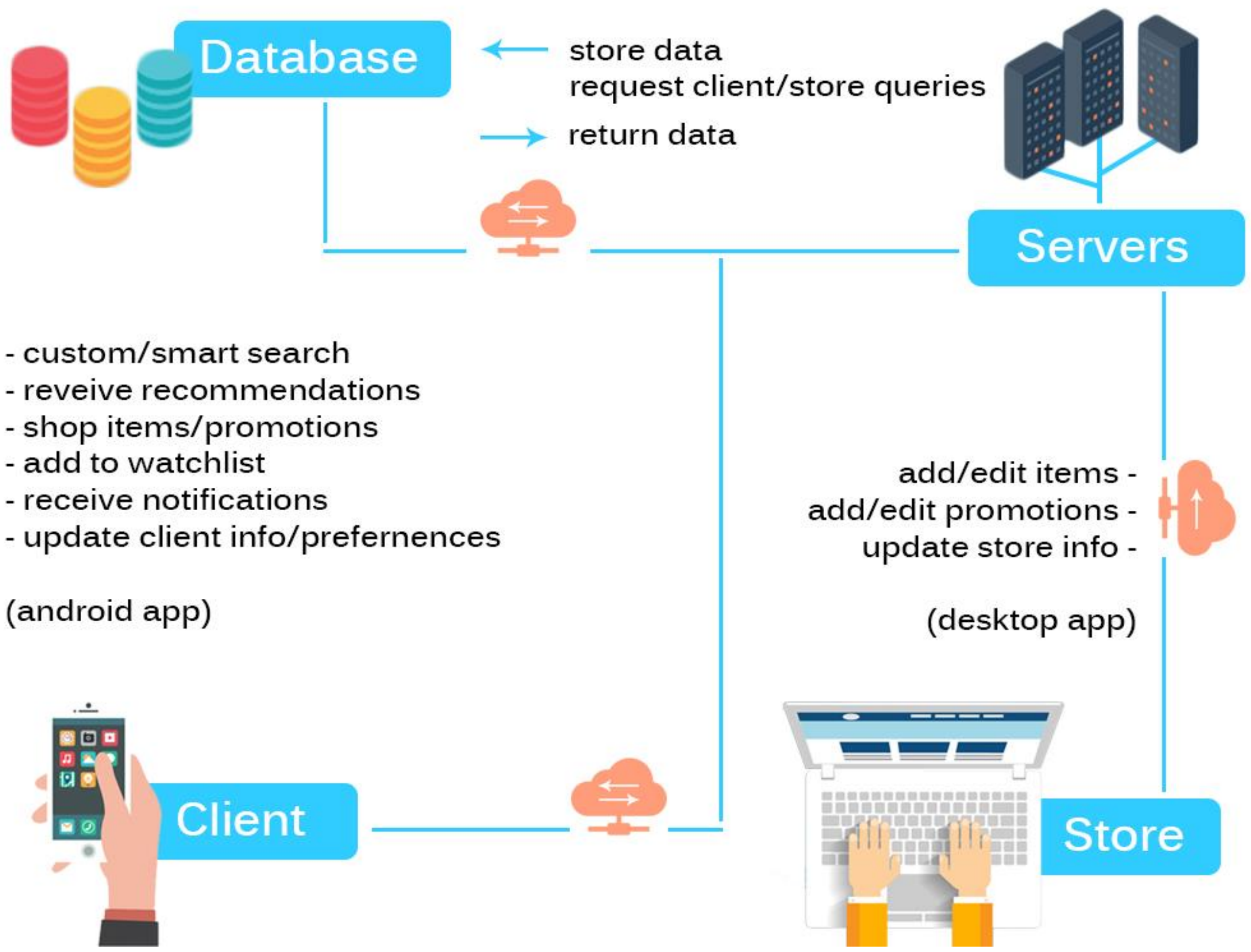


Figure 1 shopping system overall interaction

- **Users of the system**
  People of different ages interested in online shopping, product information, shopping promotions or smart suggestions based on products and shopping preferences.

- **Users' interaction with the system**
  System displays the various shopping domains which users can select from and get the available stock products with detailed information including price, manufacturer, vendor, store and other features.
  Users can input data related to their preferred items from the various shopping domains like food, clothes, sports, electronics, etc. The system will direct users to their preferred and most fitting available stock products with their detailed information.

- **Data input into the system**
  Users' Interesting items by the product world-known name or product relevant features.
  Add items and promotions to the watchlist in order to receive alerts concerning specific shopping domains, items or features.
- **Data output by the system**
  Lists of shopping domains, specific domain's items, detailed information about every item, stores and promotions.
  Suggestion based on user preferences.
  Alerts of promotions and matching preferences based on the watchlist.

2. Interaction design (client/server-side interaction blueprint)

- **Customers interaction**

  Implemented as an android app or web app or both. Includes:

  Front-end:

  - Display shopping domains, products, promotions, stores and alerts.
  - Input preferences.
  - Add, edit or remove items from the watchlist.
  - GUI (web/android) that supports the previous operations.
  - Performed by the application (web/android).

  Back-end:

  - App communicates using http protocol with the server side to perform data exchange requests and get responses then delivers them to the front-end.

  - Server interacts with customers with pre-specified privileges which are different from the store/system managers' privileges.

  - Server responds to customer requests such as prediction, searching, retrieving database information, schedule alerts and applying machine learning / recommendation algorithms.

- **Store/system managers interaction**

  Implemented as a web or desktop app to increase the usability and accuracy. Includes:

  Front-end:

  - Web/desktop app that provides data manipulation operation of adding, editing and removing items such as products, features, stores, offers, vendors, clients etc.

  Back-end:

  - App communicates with the server to perform data manipulation operation.

  - Server interacts with store/system managers with pre-specified privileges which are different from the customers' privileges.

  -Server responds to mangers' request of storing, updating or deleting data on the database.

3. Database design

- **Identifying Entities (types of information saved in the database)**
  Products, Stores, Clients, Offers/Promtions and Vendors.

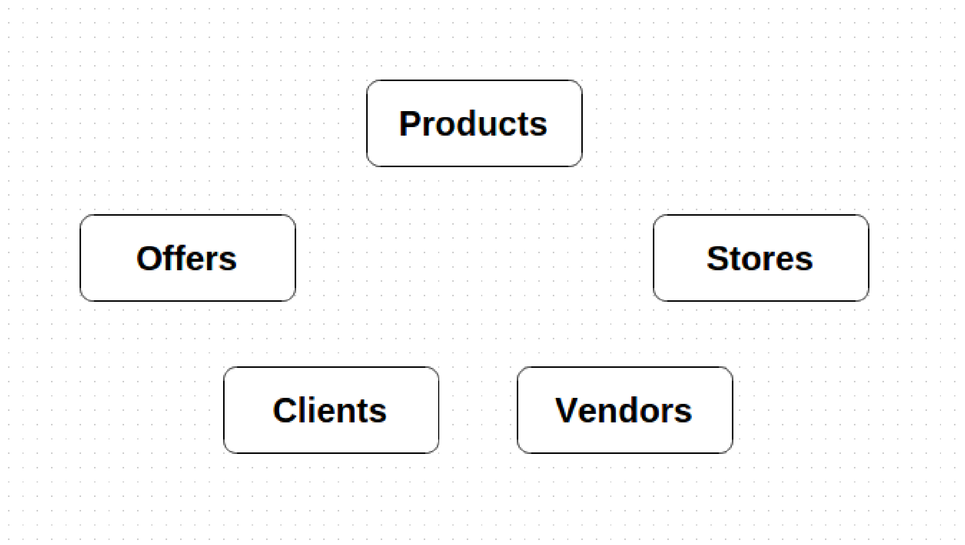


Figure 2 system entities

- **Identifying Relationships**

We have 5 choose 2 relations (5C2 = 10)

- Stores & Clients and Stores & Vendors no interest in these relations.

- **Stores & Products   M:N**

  a store may have many products (one or more) , a product may be sold in many stores (one or more).

- **Stores & Offers/Promotions   M:N**

  a store may have many offers (zero or more), an offer may exist in many stores (one or more).

- Vendors & Clients and Vendors & Offers no interest in these relations.

- **Vendors & Products   M:N**

  a vendor may supply many products (one or more), a product may be supplied by many vendors (one or more).

- **Clients & Products   1:N**

a client can prefer many products (zero or more), products are not related to clients (no interest in this backward relation).

- **Clients & Offers   1:N**

a client can prefer many offers (zero or more), offers are not related to clients (no interest in this backward relation).

- **Offers & Products   M:N**

an offer has many products (one or more), a product may be include many offers (zero or more)

1:N / 1:M one-to-many relationship
M:N many-to-many relationship

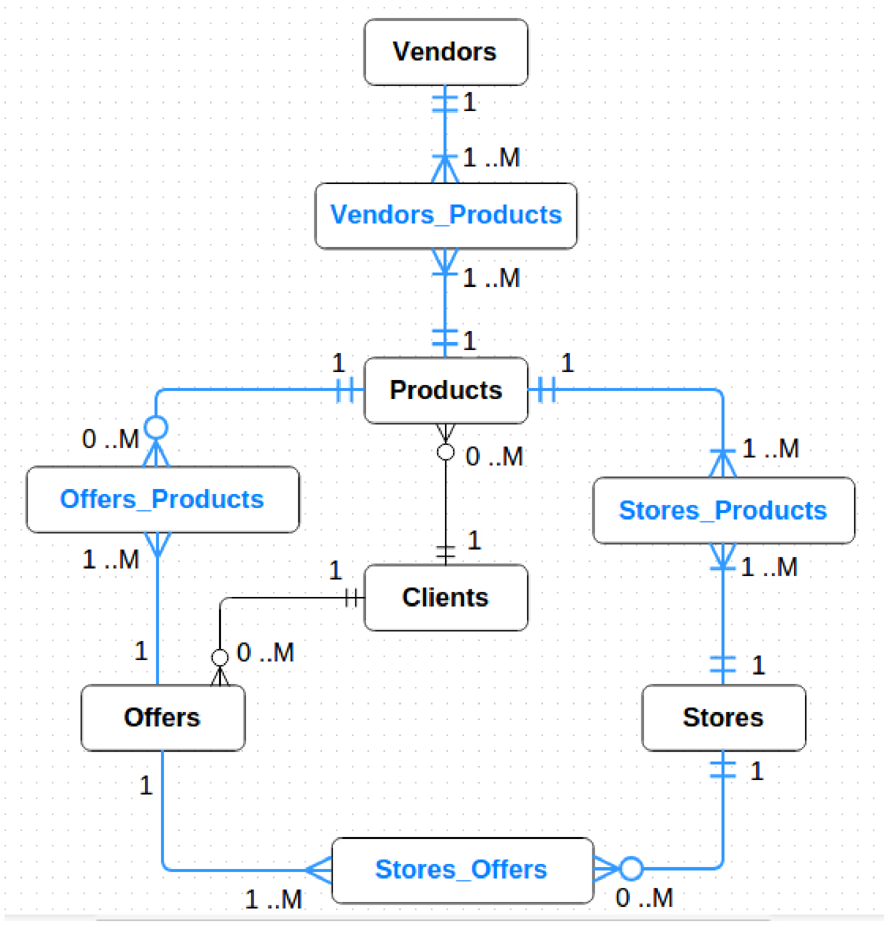


Figure 2.1 systems entities relation

- **Identifying Attributes And Specifying Attribute (column) Data Types**

  VARCHAR (Text)

  for name, address, mobile phone, telephone, email, feature columns of products and password attributes.

  The range of Length is 1 to 255 characters. VARCHAR values are sorted and compared in case-insensitive fashion

  DATETIME

  for date attributes.

  The supported range is ‘1000-01-01 00:00:00’ to ‘9999-12-31 23:59:59’. MySQL displays DATETIME values in ‘YYYY-MM-DD HH:MM:SS’ format

  Integer

  for ID attributes.

  The signed range is –2147483648 to 2147483647. The unsigned range is 0 to 4294967295.

  SMALLINT

  for quantity attribute.

  The signed range is –32768 to 32767. The unsigned range is 0 to 65535.

  Float

  for price attribute.

  Ranges are –3.402823466E+38 to –1.175494351E-38, 0 and 1.175494351E-38 to 3.402823466E+38.

4. Databse realization

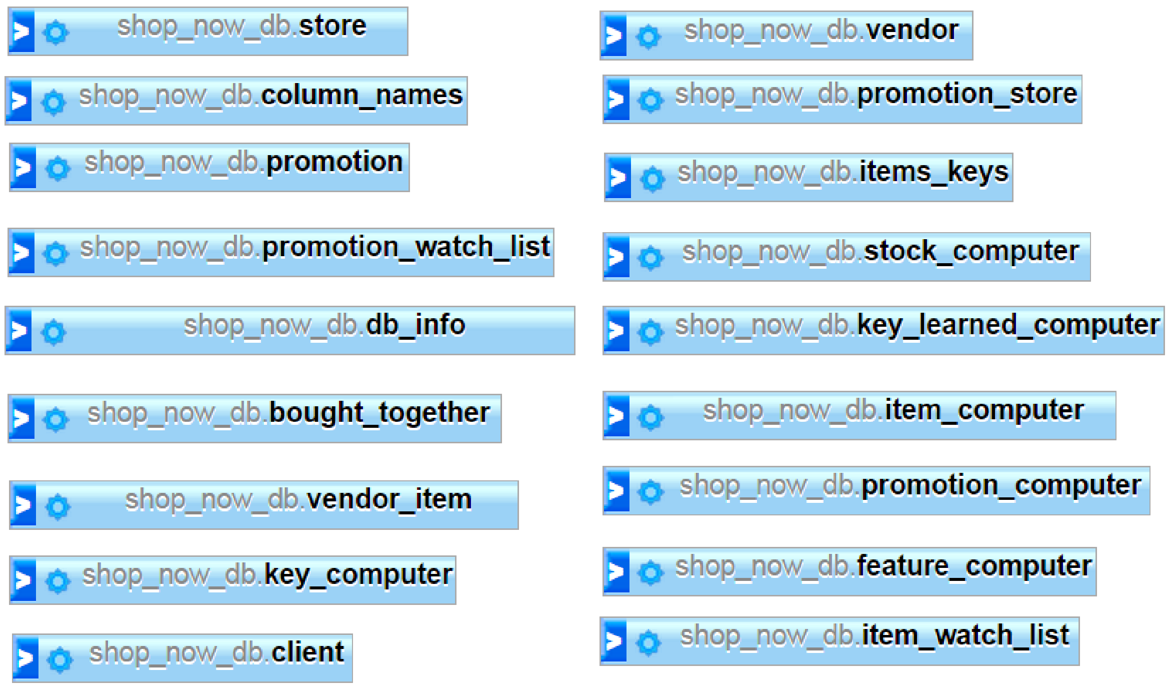

shop_now_db.store
shop_now_db.column_names
shop_now_db.promotion
shop_now_db.promotion_watch_list
shop_now_db.db_info
shop_now_db.bought_together
shop_now_db.vendor_item
shop_now_db.key_computer
shop_now_db.client
shop_now_db.vendor
shop_now_db.promotion_store
shop_now_db.items_keys
shop_now_db.stock_computer
shop_now_db.key_learned_computer
shop_now_db.item_computer
shop_now_db.promotion_computer
shop_now_db.feature_computer
shop_now_db.item_watch_list

Figure 3 system database tables

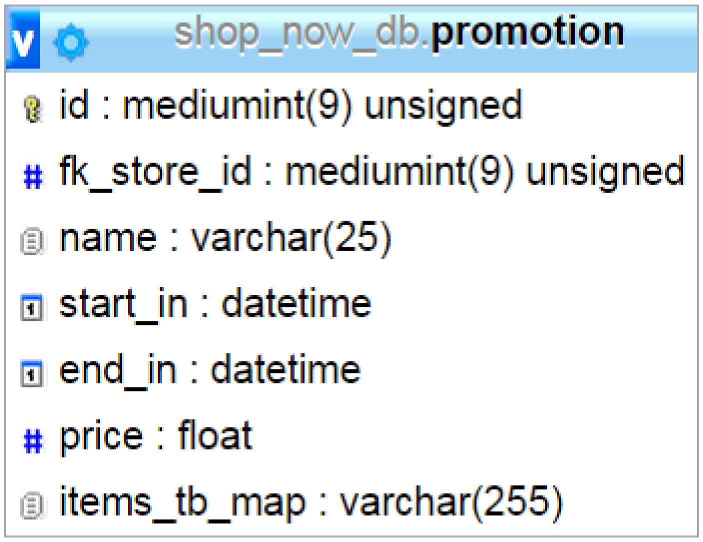
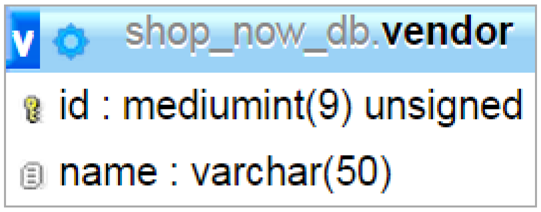

shop_now_db.promotion
id : mediumint(9) unsigned
fk_store_id : mediumint(9) unsigned
name : varchar(25)
start_in : datetime
end_in : datetime
price : float
items_tb_map : varchar(255)

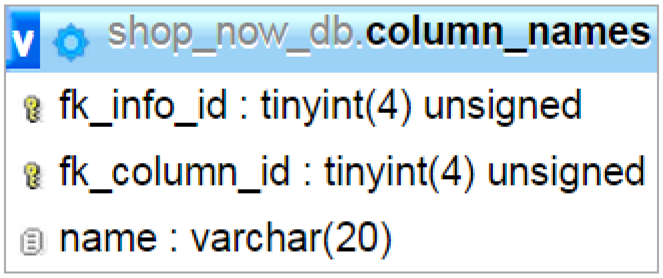
Figure 3.1 promotion table diagram



shop_now_db.vendor

id : mediumint(9) unsigned
name : varchar(50)

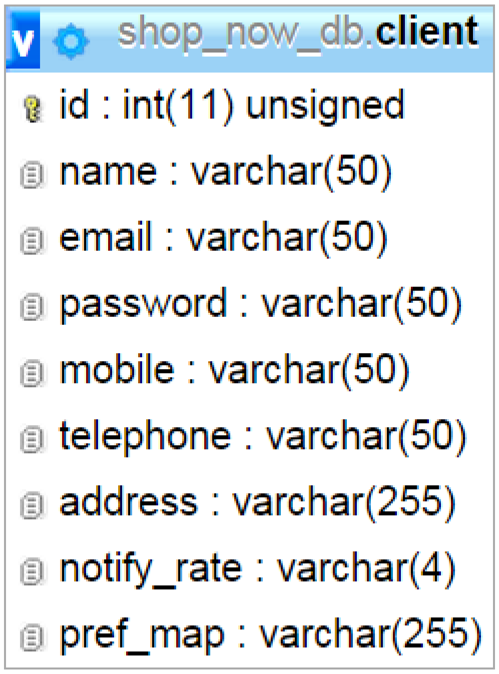
Figure 3.2 vendor table diagram



shop_now_db.column_names

fk_info_id : tinyint(4) unsigned
fk_column_id : tinyint(4) unsigned
name : varchar(20)

Figure 3.3 column names table diagram



shop_now_db.client

id : int(11) unsigned
name : varchar(50)
email : varchar(50)
password : varchar(50)
mobile : varchar(50)
telephone : varchar(50)
address : varchar(255)
notify_rate : varchar(4)
pref_map : varchar(255)

Figure 3.4 client table diagram

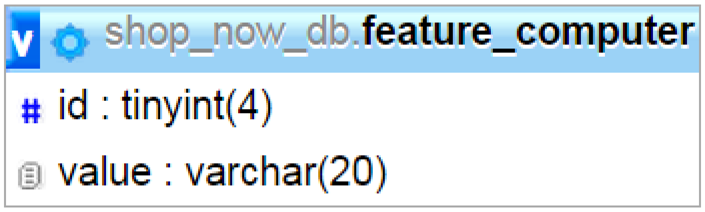


Figure 3.5 feature table diagram

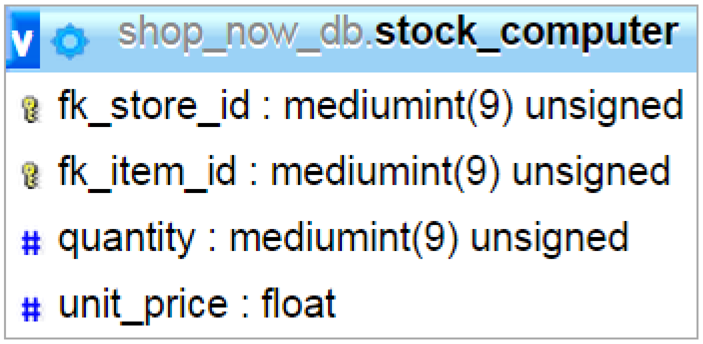

shop_now_db.stock_computer
fk_store_id : mediumint(9) unsigned
fk_item_id : mediumint(9) unsigned
quantity : mediumint(9) unsigned
unit_price : float

Figure 3.6 stock table (computer category) diagram

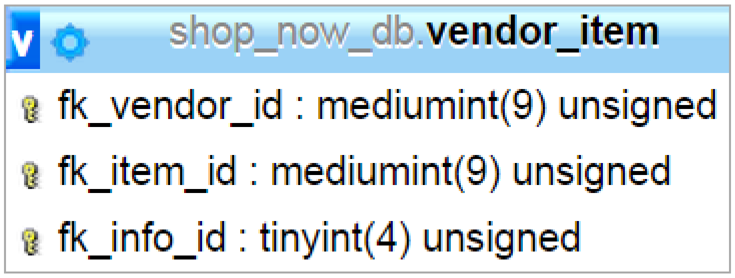

shop_now_db.vendor_item
fk_vendor_id : mediumint(9) unsigned
fk_item_id : mediumint(9) unsigned
fk_info_id : tinyint(4) unsigned

Figure 3.7 vendor item mapping table diagram

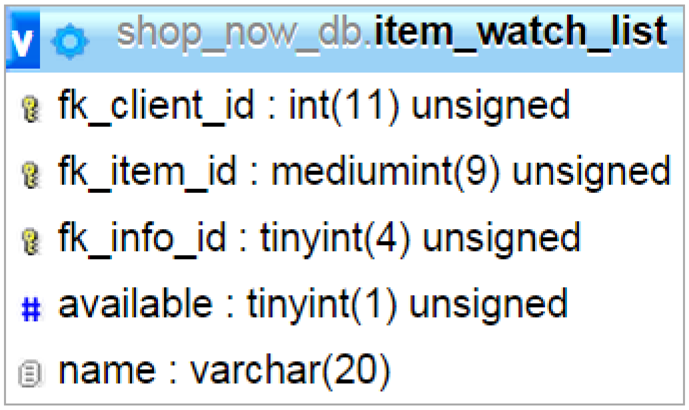
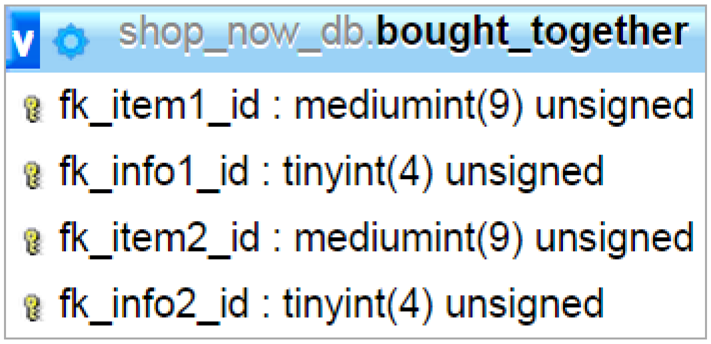
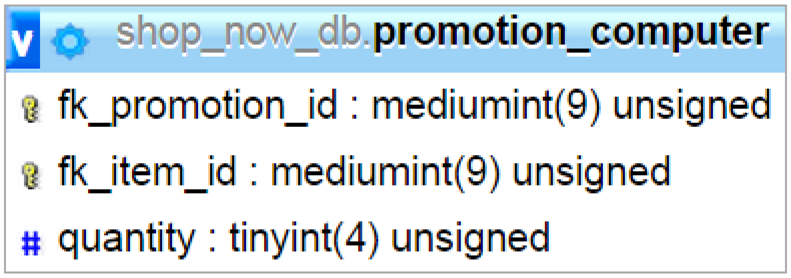

shop_now_db.item_watch_list
fk_client_id : int(11) unsigned
fk_item_id : mediumint(9) unsigned
fk_info_id : tinyint(4) unsigned
available : tinyint(1) unsigned
name : varchar(20)

Figure 3.8 item watch list table diagram


shop_now_db.**bought_together**

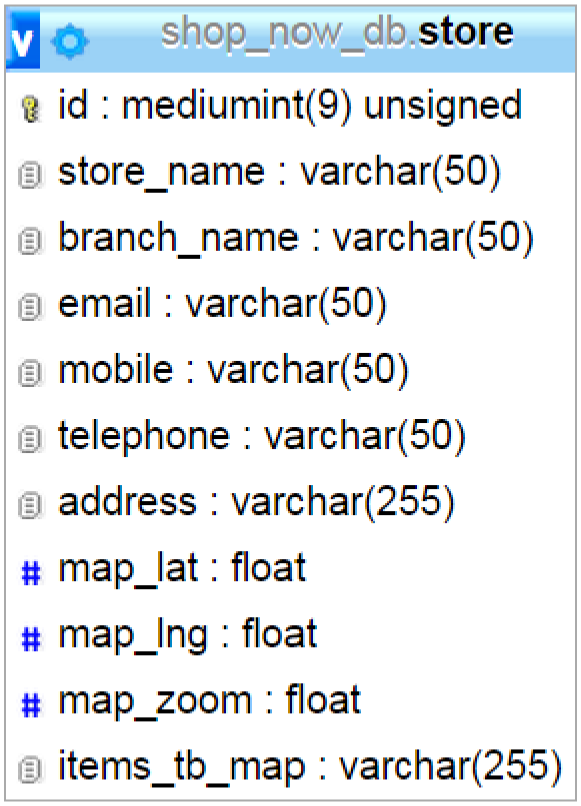
fk_item1_id : mediumint(9) unsigned
fk_info1_id : tinyint(4) unsigned
fk_item2_id : mediumint(9) unsigned
fk_info2_id : tinyint(4) unsigned

Figure 3.9 items bought together table diagram


shop_now_db.**promotion_computer**

fk_promotion_id : mediumint(9) unsigned
fk_item_id : mediumint(9) unsigned
quantity : tinyint(4) unsigned

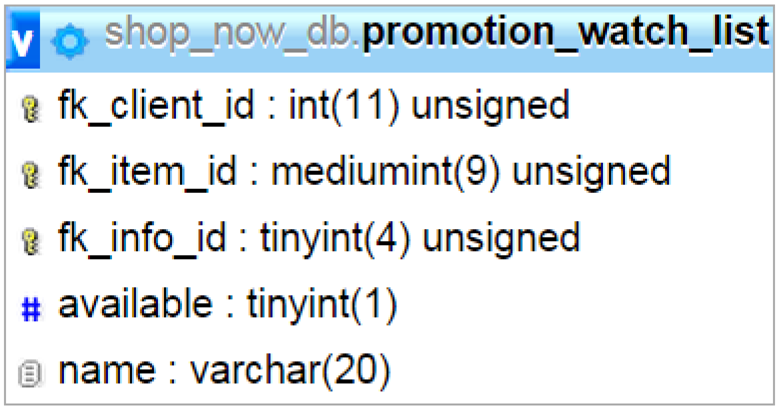
Figure 3.10 promotion table (computer category) diagram



shop_now_db.**store**

```
id : mediumint(9) unsigned
store_name : varchar(50)
branch_name : varchar(50)
email : varchar(50)
mobile : varchar(50)
telephone : varchar(50)
address : varchar(255)
map_lat : float
map_lng : float
map_zoom : float
items_tb_map : varchar(255)
```

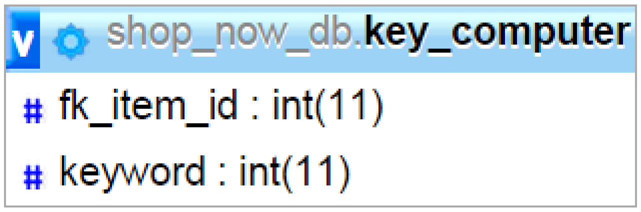
Figure 3.11 store table diagram



shop_now_db.**promotion_watch_list**

```
fk_client_id : int(11) unsigned
fk_item_id : mediumint(9) unsigned
fk_info_id : tinyint(4) unsigned
available : tinyint(1)
name : varchar(20)
```

Figure 3.12 promotion watch list table diagram

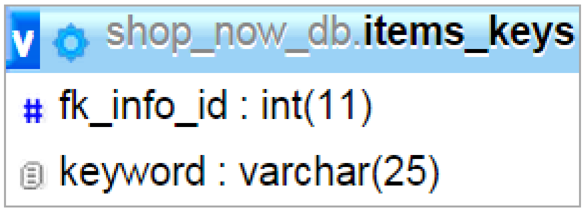


shop_now_db.**key_computer**

fk_item_id : int(11)

keyword : int(11)

Figure 3.13 key computer table diagram

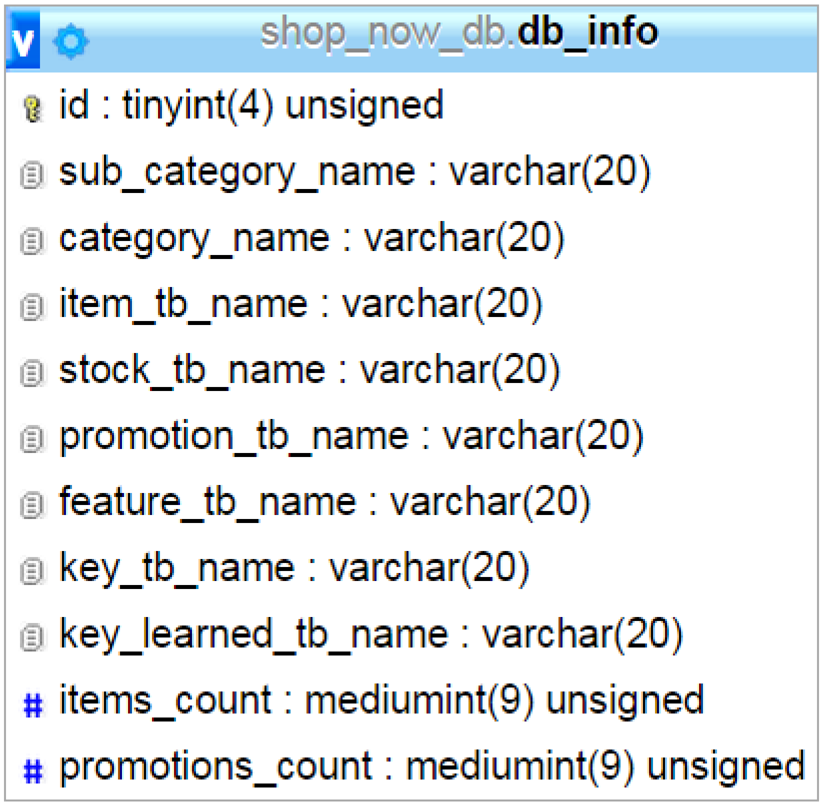


shop_now_db.**items_keys**

fk_info_id : int(11)

keyword : varchar(25)

Figure 3.14 items keys table diagram



shop_now_db.**db_info**

id : tinyint(4) unsigned

sub_category_name : varchar(20)

category_name : varchar(20)

item_tb_name : varchar(20)

stock_tb_name : varchar(20)

promotion_tb_name : varchar(20)

feature_tb_name : varchar(20)

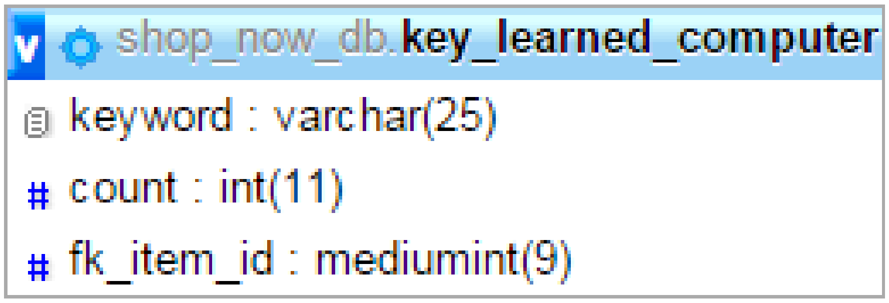
key_tb_name : varchar(20)

key_learned_tb_name : varchar(20)

items_count : mediumint(9) unsigned

promotions_count : mediumint(9) unsigned

Figure 3.15 database info table diagram



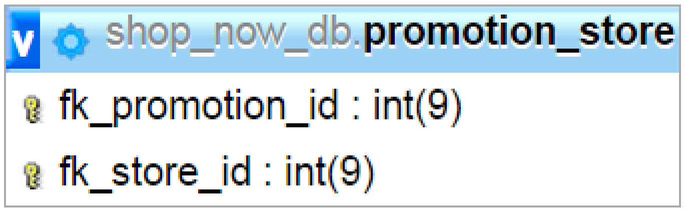
shop_now_db.key_learned_computer

keyword : varchar(25)

count : int(11)

fk_item_id : mediumint(9)

Figure 3.16 key learned for computer table diagram

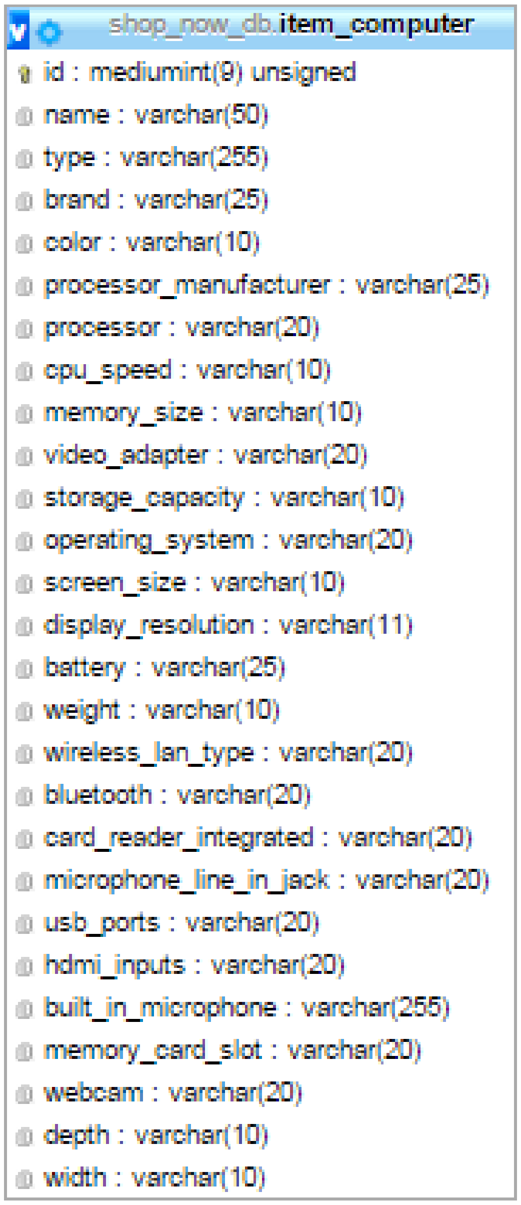


shop_now_db.promotion_store

fk_promotion_id : int(9)

fk_store_id : int(9)

Figure 3.17 promotion store table diagram



shop_now_db.item_computer

```
id : mediumint(9) unsigned
name : varchar(50)
type : varchar(255)
brand : varchar(25)
color : varchar(10)
processor_manufacturer : varchar(25)
processor : varchar(20)
cpu_speed : varchar(10)
memory_size : varchar(10)
video_adapter : varchar(20)
storage_capacity : varchar(10)
operating_system : varchar(20)
screen_size : varchar(10)
display_resolution : varchar(11)
battery : varchar(25)
weight : varchar(10)
wireless_lan_type : varchar(20)
bluetooth : varchar(20)
card_reader_integrated : varchar(20)
microphone_line_in_jack : varchar(20)
usb_ports : varchar(20)
hdmi_inputs : varchar(20)
built_in_microphone : varchar(255)
memory_card_slot : varchar(20)
webcam : varchar(20)
depth : varchar(10)
width : varchar(10)
```

Figure 3.18 item computer table diagram

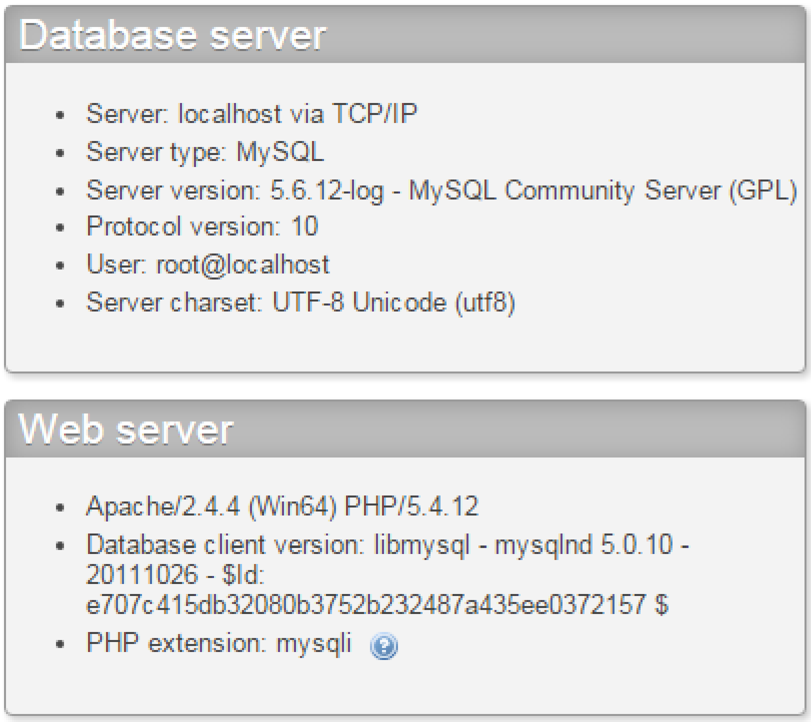
5. Algorithms

- **Custom Search Algorithm**
  - Matches the required features against all items stored at the specified subcategory table.
  - Requires linear scan of all items and fixed time for the exact feature matching (number of feature).
  - Assigns a zero rank to each scanned item. Rank is incremented by one if the item matches a required feature. Item is rejected if it does not match a single required feature.
  - Sorts the result items set according to their respected rank.
- **Smart Search Algorithm**
  - Parse user’s input to get keywords.
  - Applys two level keyword matching, the first matches keywords with subcategories and selects the highest rank subcategory (increment a subcategory’s rank by one if it matches a keyword).
  - System holds all the keywords that mapped to the hightest rank subcategory and match them with the items stored in this subcategory and selects the highest rank item.
  - Systems recommends the hightest rank item based on the two level keyword matching
  - System stores all the unmapped/uncaught keywords in the learned keys table which stores the unknown keys found in user’s input that could not be mapped during a smart search sessions with their final subcategory and item search result.
  - Systems perform periodic checks over the learned keys to see if they have been repeated by users over time. If keys match a prespecified criterion, system adds them to the system keywords.

6. System evalaution (Performance Study)

System was tested with

- **1.37 GB RAM**
- **Intel i5 2.4 GHz**
- **Apache Server**
- **MySQL engine**



Database server

- Server: localhost via TCP/IP
- Server type: MySQL
- Server version: 5.6.12-log - MySQL Community Server (GPL)
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.4.4 (Win64) PHP/5.4.12
- Database client version: libmysql - mysqlnd 5.0.10 - 20111026 - $Id: e707c415db32080b3752b232487a435ee0372157 $
- PHP extension: mysqli

Figure 4 local server properties

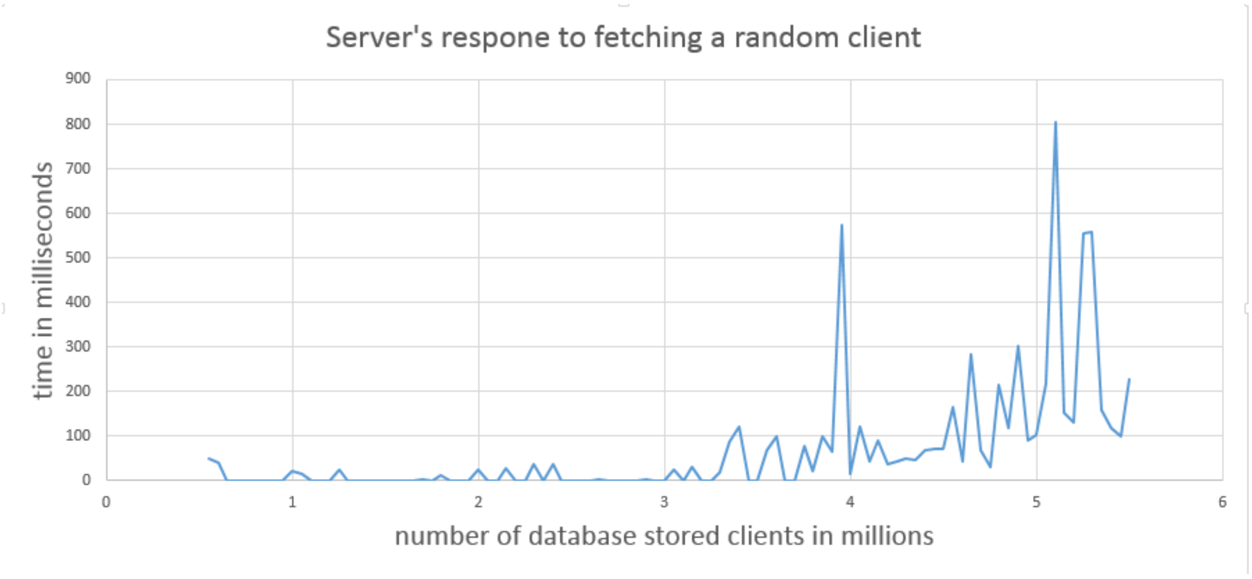


Figure 4.1

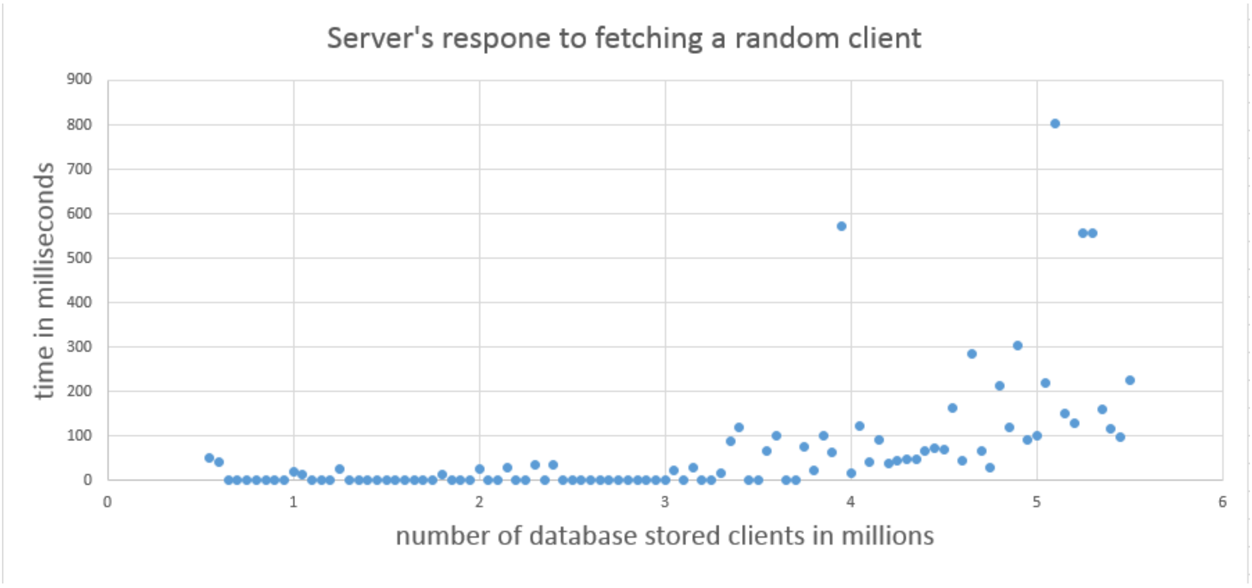


Figure 4.2

Figure 4.1 and 4.2 illustrate server's response time to fetching a client chosen at random from clients table versus the total number of stored clients.
The total number of clients ranges 0.5 - 5.5 millions. Server's response tends to cluster around 0 millisconds with 3 million clients or less.
Reaching 3 millions, response clusters around 100 milliseconds.
Reaching 4.5 millions, response clusters around 200 milliseconds.

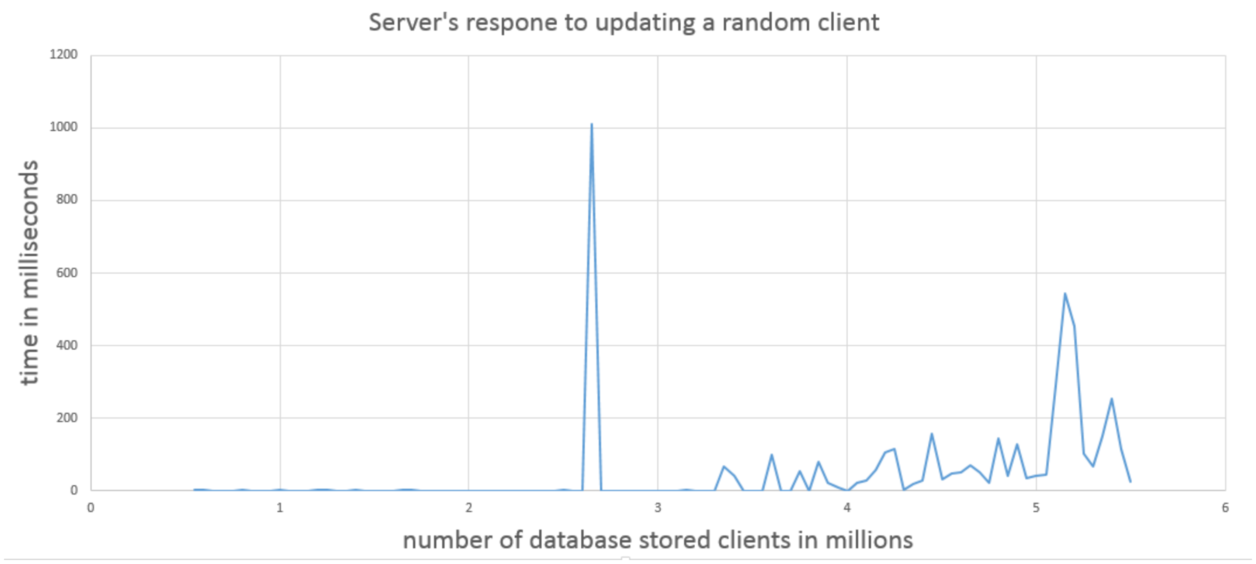


Figure 4.3

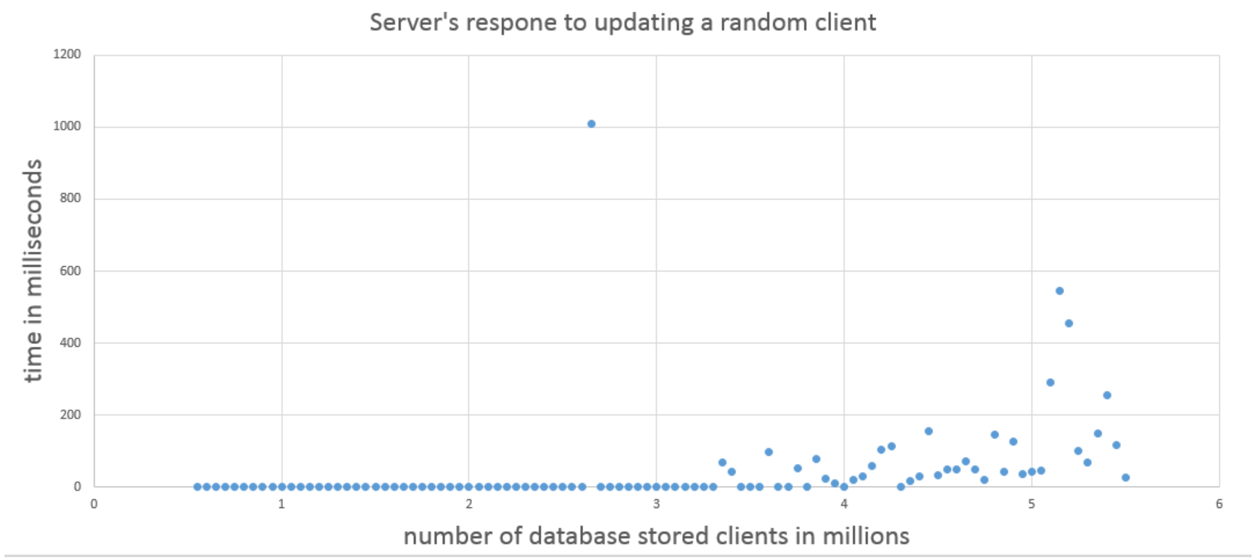


Figure 4.4

Figure 4.3 and 4.4 illustrate server's response time to updating client information chosen at random from clients table versus the total number of stored clients. The total number of clients ranges 0.5 – 5.5 millions. Server's response tends to cluster around 0 milliscонds with 3 million clients or less.
Reaching 3 millions, response clusters around 100 milliseconds.

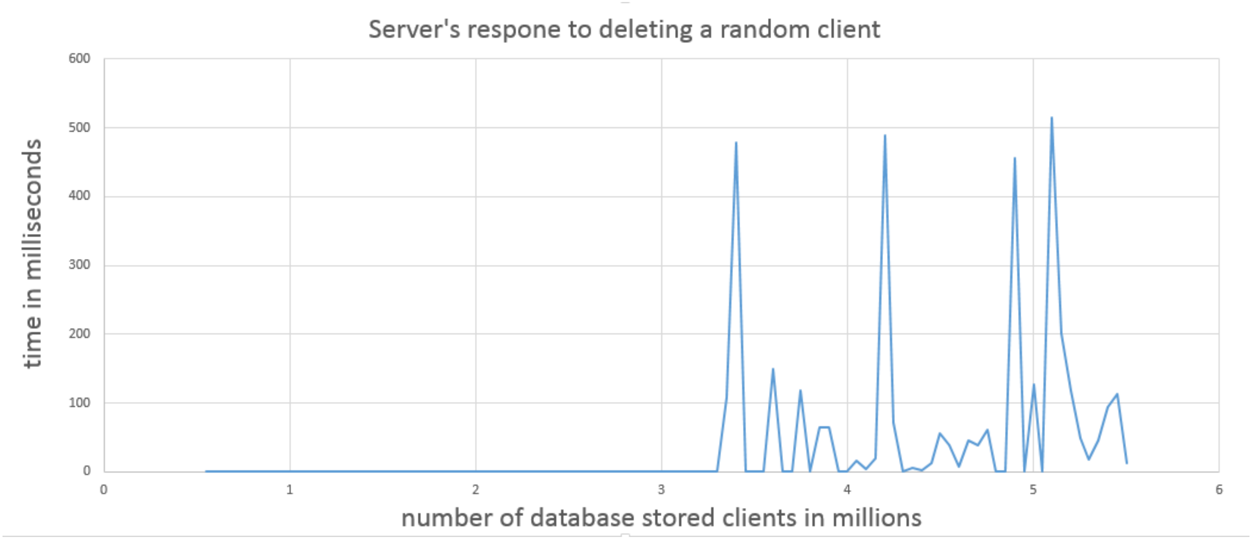


Figure 3.5

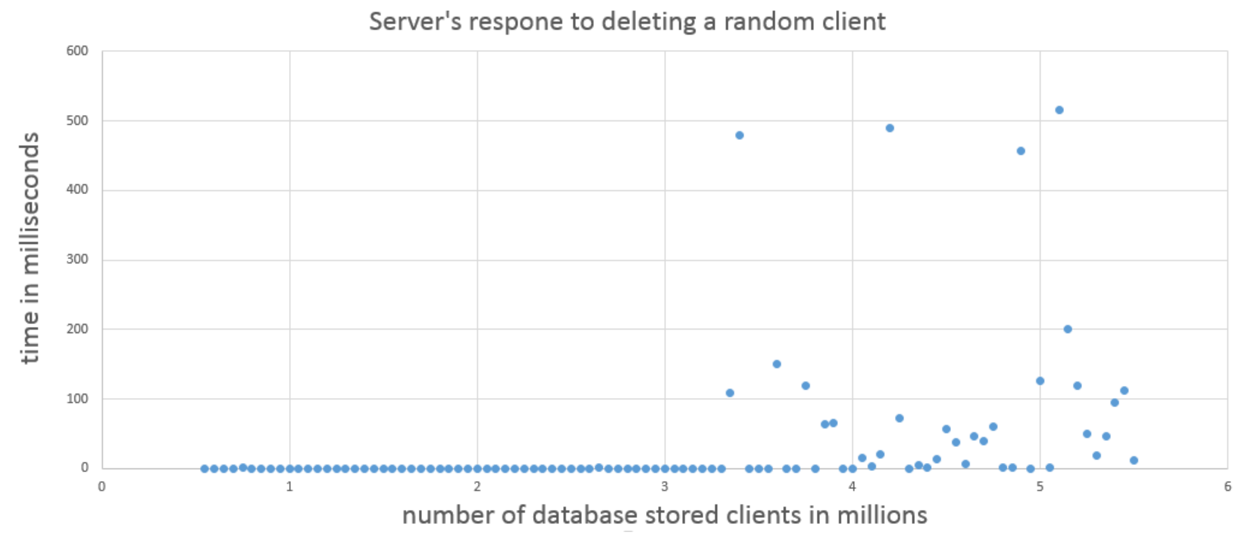


Figure 4.6

Figure 4.5 and 4.6 illustrate server's response time to deleting a client chosen at random from clients table versus the total number of stored clients.
The total number of clients ranges 0.5 – 5.5 millions. Server's response tends to cluster around 0 millisconds with 3 million clients or less.
Reaching 3 millions, response clusters around 100 milliseconds.
Deleting a client requires more time than fetching as deletion affects the primiary key and thus affects table indexing.

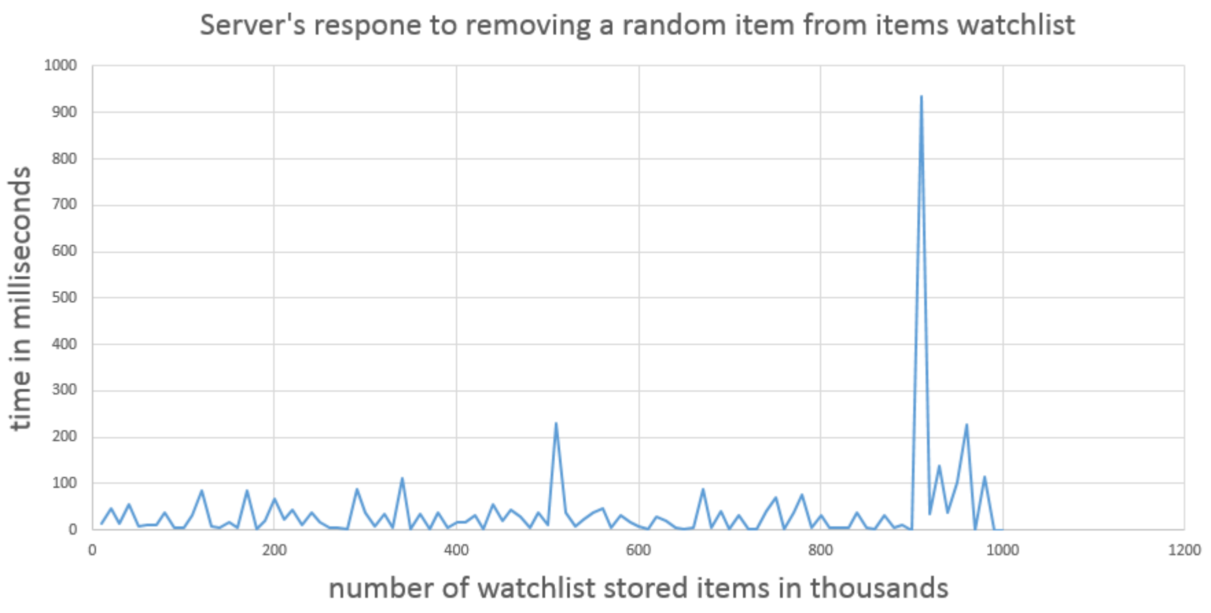


Figure 4.7

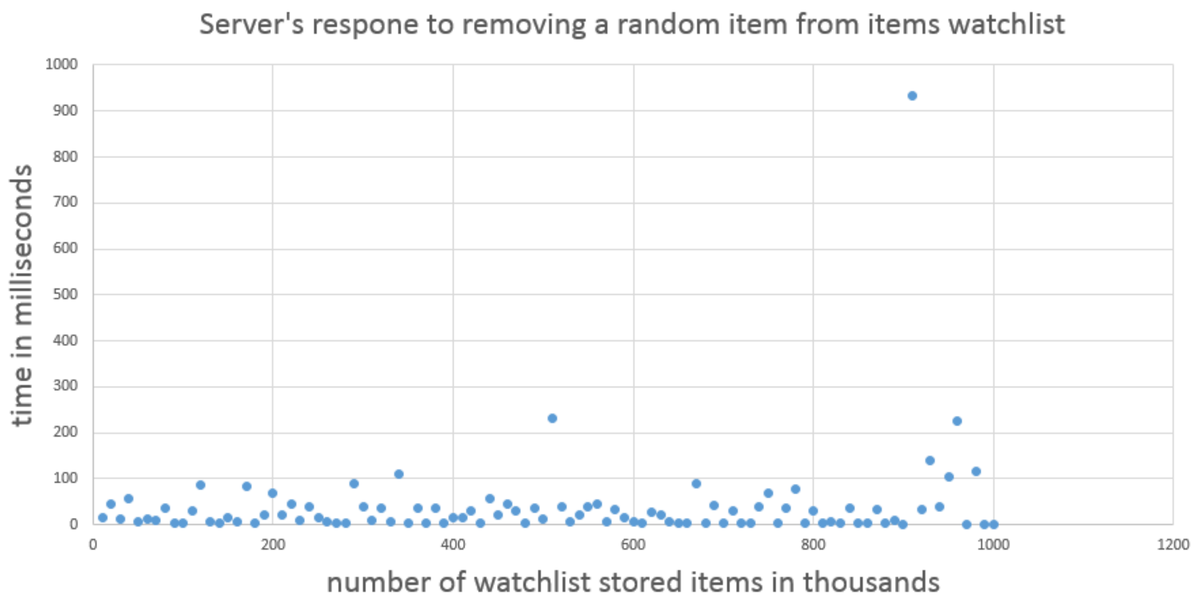


Figure 4.8

Figure 4.7 and 4.8 illustrate server's response time to removing an item chosen at random from items watch list table versus the total number of watch list items. The total number of watch list items ranges 0 – 1 million. Server's response tends to cluster around 50 millisconds.

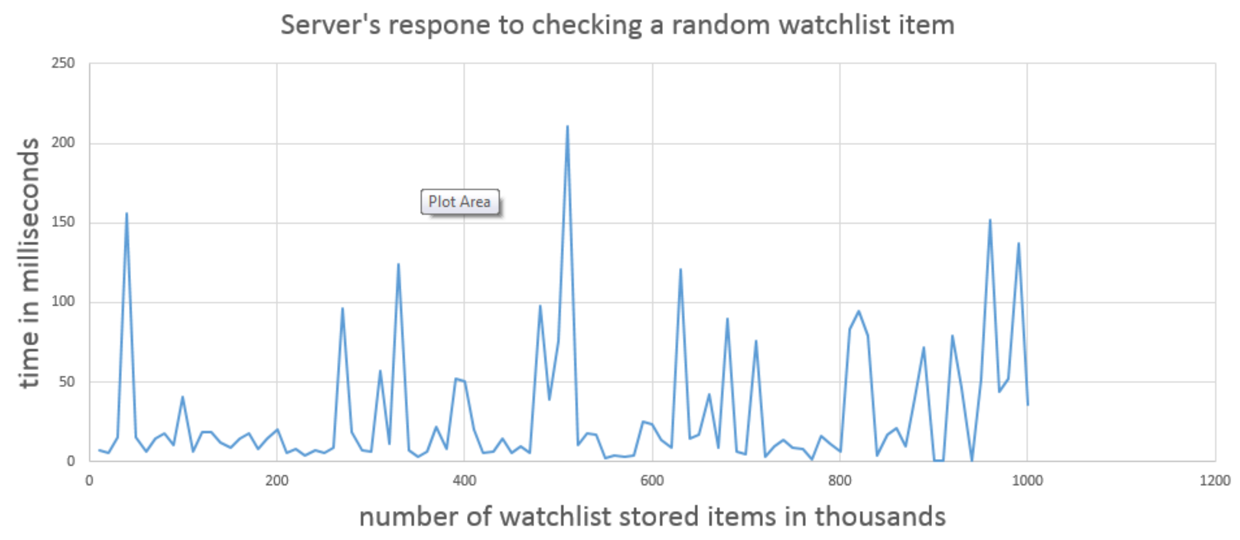


Figure 4.9

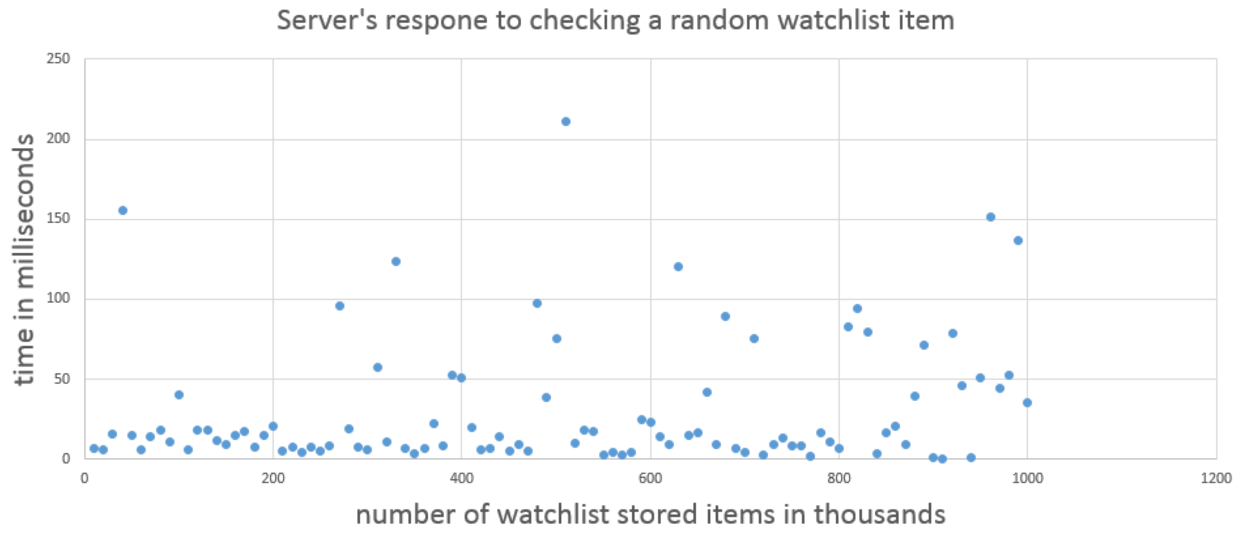


Figure 4.10

Figure 4.9 and 4.10 illustrate server's response time to checking the status of an item chosen at random from watch list table versus the total number of watch list items. The total number of watch list items ranges 0 – 1 million. Server's response tends to cluster under 25 millisconds and around 75 milliseconds with some glitches in between.

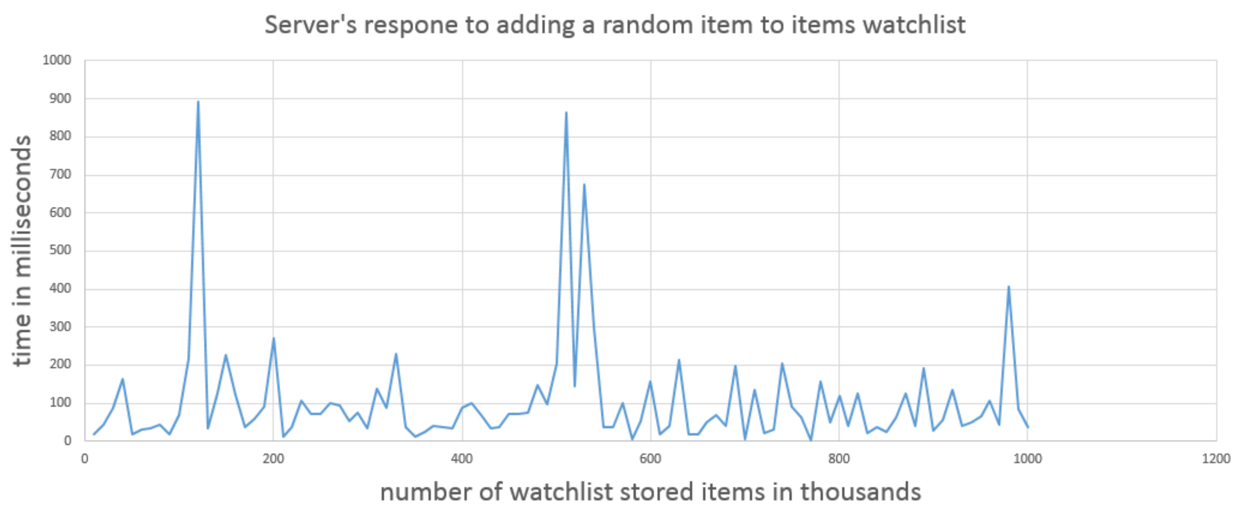


Figure 4.11

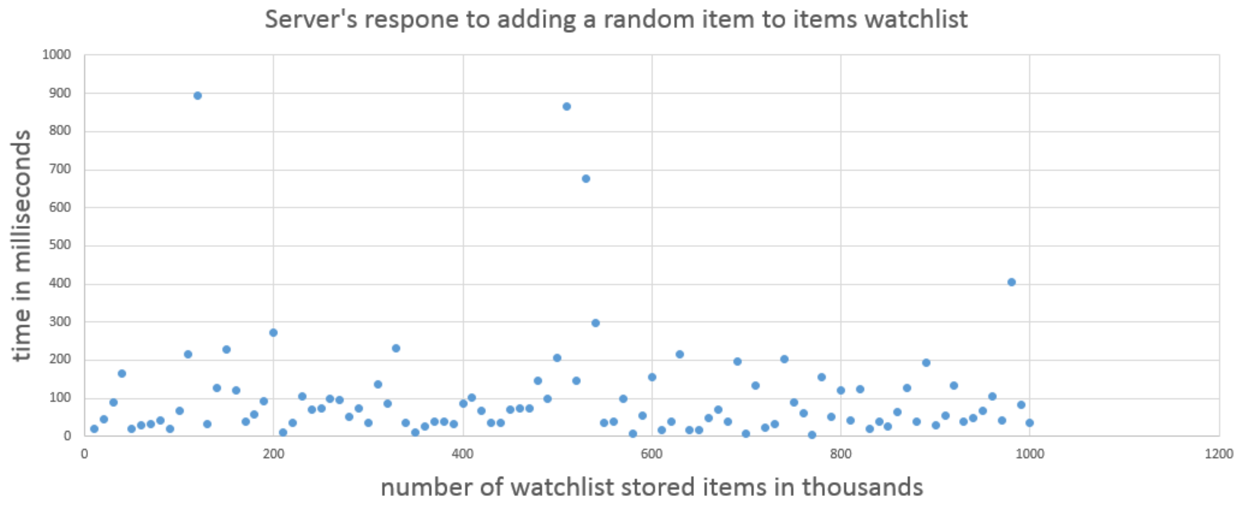


Figure 4.12

Figure 4.11 and 4.12 illustrate server's response time to adding a random item to watch list items table versus the total number of watch list items.
The total number of existing watch list items ranges 0 – 1 million. Server's response tends to cluster around 100 millisconds.

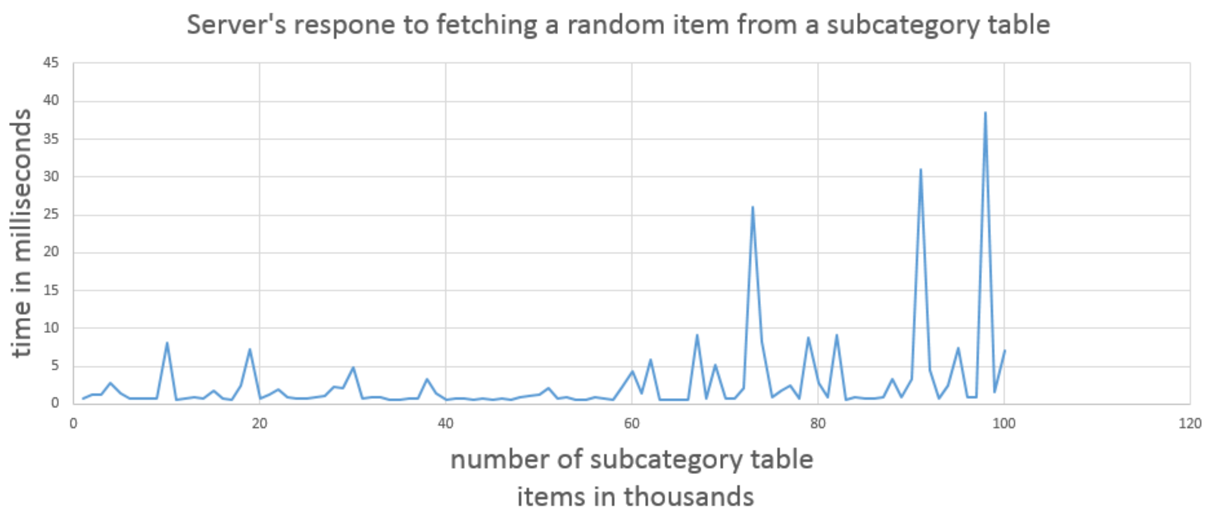


Figure 4.13

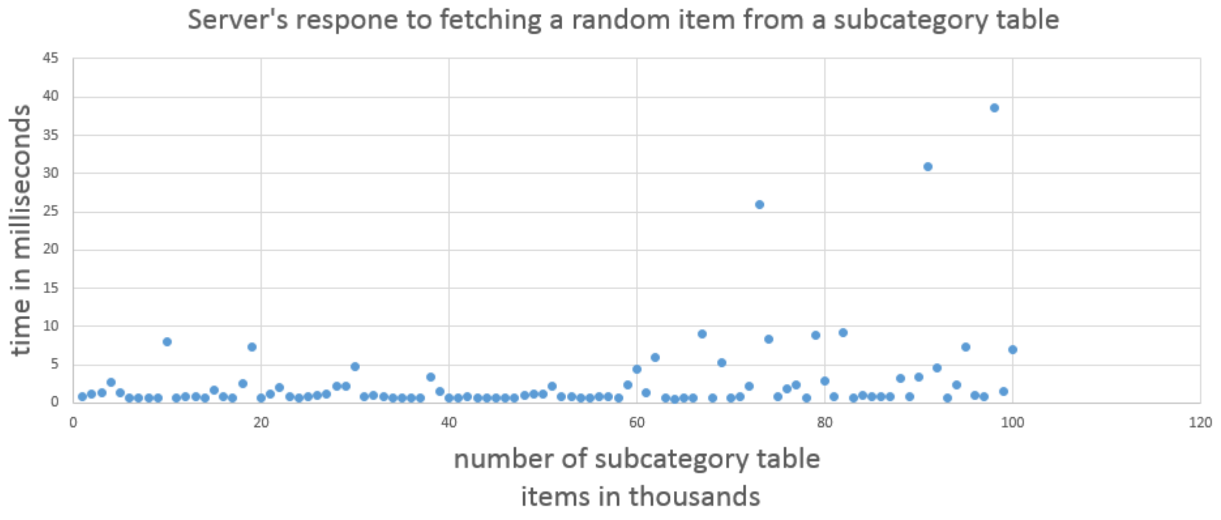


Figure 4.145

Figure 4.13 and 4.14 illustrate server's response time to fetching an item chosen at random from items table versus the total number of stored items in this subcategory table.

The total number of items in this subcategory ranges 0 – 100k. Server's response tends to cluster around 2.5 millisconds with 60k items or less. Reaching 60k, response clusters around 5 milliseconds.

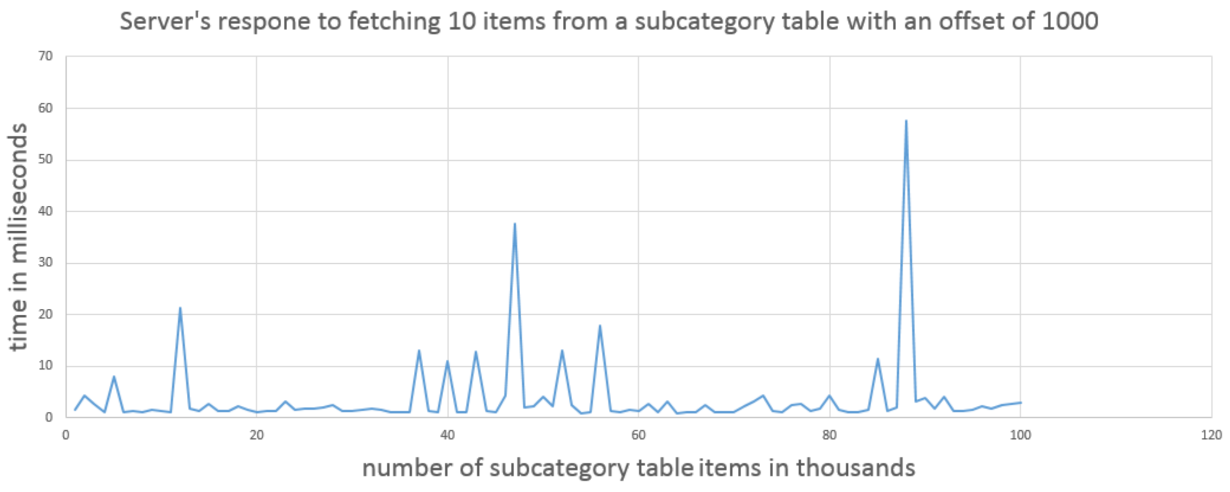


Figure 4.15

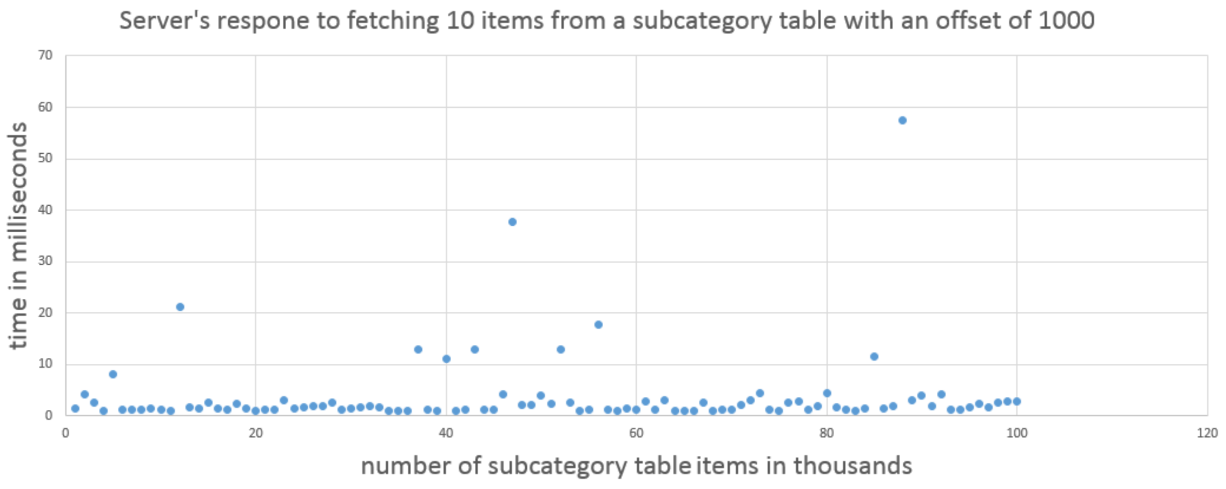


Figure 4.16

Figure 4.15 and 4.16 illustrate server's response time to fetching 10 items chosen from items table after skipping 1000 items **(simulating the situation where a client has viewed 1000 items and asks for 10 items more by pagination)** versus the total number of stored items in this subcategory table.

The total number of items in this subcategory ranges 0 – 100k. Server's response tends to cluster under 5 millisconds with some glitches in between.

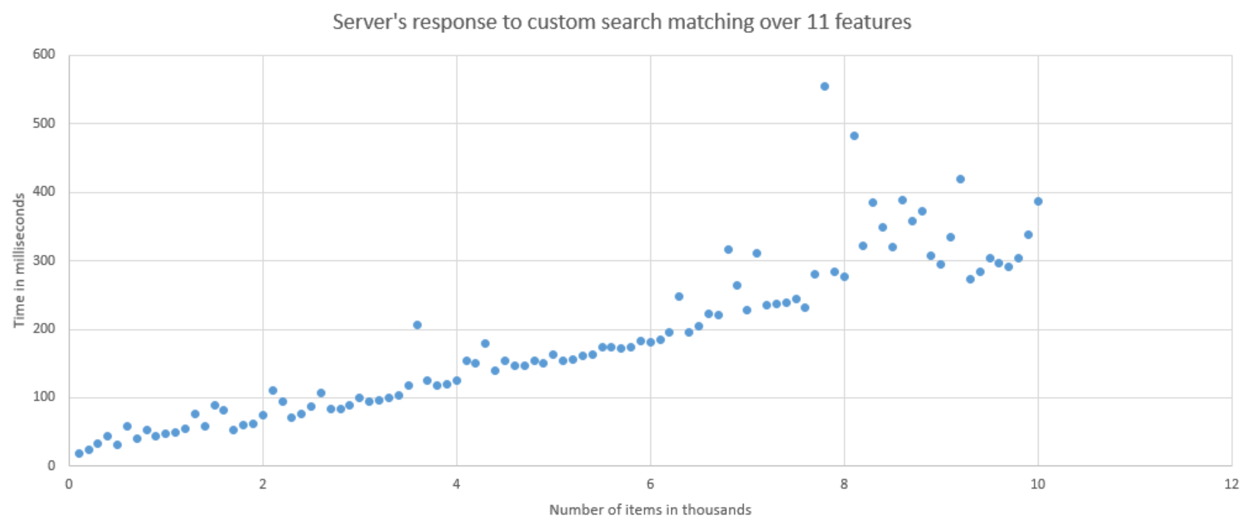


Figure 4.17

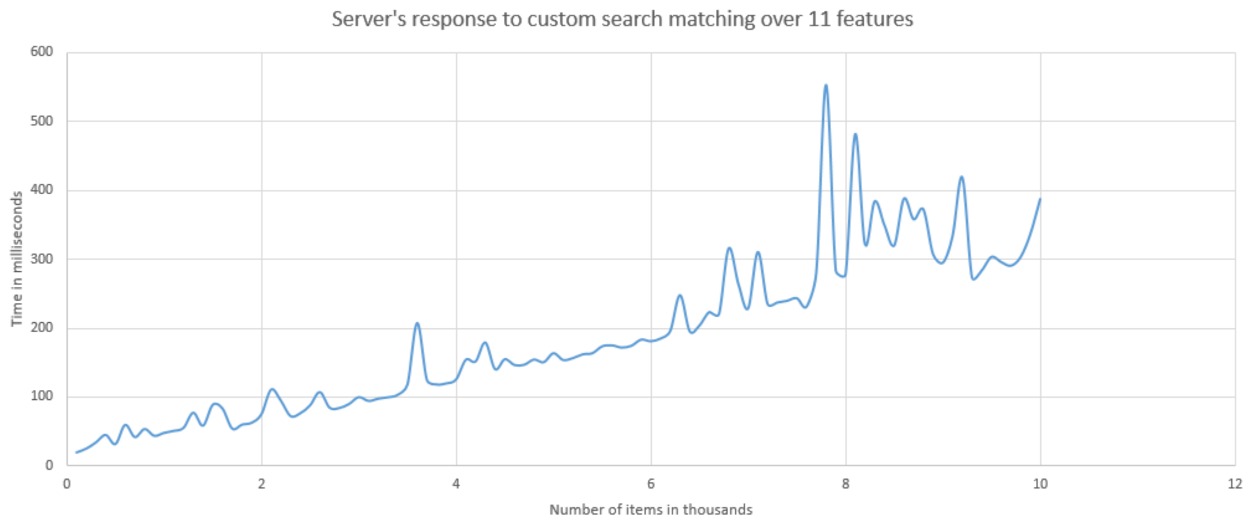


Figure 4.18

Figure 4.17 and 4.18 illustrate the optimization in the server's response time to match required features (11 features) with items table versus the total number of stored items in this subcategory table.

The total number of items in this subcategory ranges 0 – 10,000.