INFSCI 2710 Final Project Report (Spike System)

- Team member:
  - Yiming Zeng, Qisheng Ye, Xizhi Wu
  - yiz182@pitt.edu, qiy48@pitt.edu, xiw183@pitt.edu

- Course: Database Management
- Instructor: Babichenko, Dmitriy
- Date: April 23rd, 2022

- Table of contents

- ## Introduction

Spike system is common marketing for e-commerce websites like Amazon, TaoBao, and eBay, doing clearance cell on a small number of commodities at a limited time. So as to get a good bargain, users will flock to the website before the spike activity starts and the check button will receive thousands of clicks within seconds, resulting in the system facing challenges. This system is a high concurrency time-limited rush buying and second killing system developed by SpringBoot. In addition to the basic functions of login, viewing commodity list, second killing, and placing orders, the project also realizes system caching, degradation, and current restriction for high concurrency.

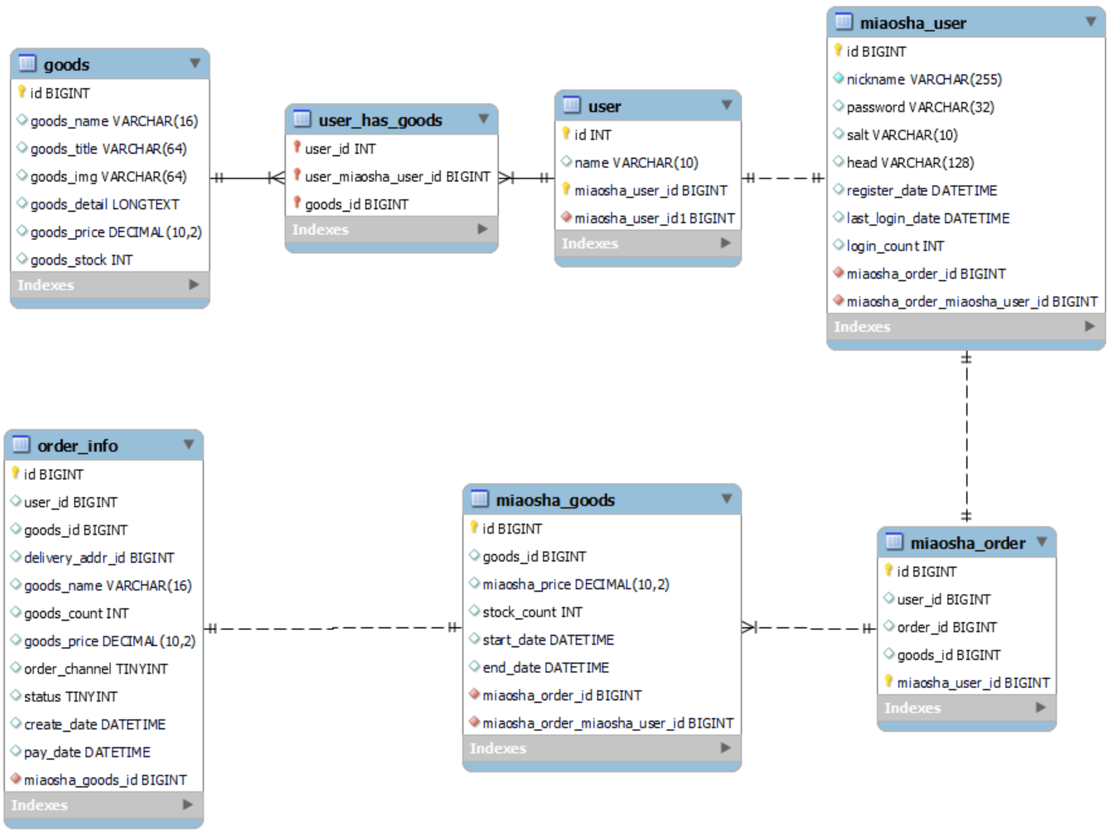
Below is the technology used by the spike system: Front end technology (bootstrap, jQuery, thyme leaf); Back end technology (spring boot, Mybatis, MySQL); Middleware technology (Druid, Redis, Rabbitmq, guava).

All online shopping users are the target audience of this system. In large-scale promotional activities held by merchants, users may rush to buy limited goods. On the one hand, this system reduces the pressure on the platform, on the other hand, it also improves the user experience.

The database stores a lot of important information such as goods, orders, and users. In this project, we also use MD5 encryption technology. The password entered by the user and the fixed salt is encrypted through MD5 to generate the password after the first encryption. Then, the password and the randomly generated salt are encrypted for the second time through MD5. Finally, the password after the second encryption and the first fixed salt is stored in the database. The first encryption can prevent the user's plaintext password from being transmitted on the network, while the second encryption can prevent the database from being stolen, avoid anti-pushing the password through MD5, and realize double insurance.

We also deployed Redis and Rabbitmq in our system to ensure concurency. Redis is a no sql database that can handle situation when traditional relational database cannot. When we need to access our mysql database, we access Redis first to check for our stock. If there is not enough stock, we will directly return this request and save more memory and resources for our system. And when user are making orders asynchronously, we first store them in a Rabbitmq queue, and then return to our server. That will give our users better experience and be more impartial to all Spike System user.

- UML-compliant E-R Model



- Business rules(miaosha in Chinese means Buying within seconds)

Entity 1	**Entity 2**	**Cardinality on Entity 1 side**	**Cardinality on Entity 2 side**	**Business Rule(s)**
goods	user	1..*	1..*	A user may buy multiple kinds of goods, each kind of good can be purchased by multiple users given enough items in stock.
user	miaosha_user	1..*	1..*	A user is a miaosha_user. A miaosha_user is a user.
miaosha_user	miaosha_order	1..*	1..*	Each miaosha_user can only make one miaosha_order at one sales promotion. And each miaosha_order can only be created by one user.

miaosha_order	miaosha_goods	1..*	1..*	Each miaosha_order only contains one kind of miaosha_goods. If there is enough stock, each kind of miaosha_goods can be in multiple miaosha_order
miaosha_goods	order_info	1..*	1..*	A miaosha_goods only have a kind order_info, while a order_order only have a kind miaosha_goods.

- Attribute descriptions

Goods		
id(PK)	BIGINT	Description
goods_name	VARCHAR	The name of goods
goods_title	VARCHAR	The type of goods
goods_img	VARCHAR	The pictures of goods
goods_detail	VARCHAR	Goods detail
goods_price	DECIMAL	Goods price
goods_stock	INT	The number of left goods

user_has_goods		
user_id(FK)	INT	The user id
user_good_id	BIGINT	Combined user_id and goods_id
goods_id(FK)	BIGINT	The goods id

user		

id(PK)	INT	ID for identifying user
name	VARCHAR(10)	Name of user
miaosha_user_id(PK)	BIGINT	ID for identifying users who join in miaosha sales promotion

miaosha_user		
id(PK)	BIGINT	id
nickname	Data Type	nickname
password	VARCHAR	password of miaosha_user
salt	VARCHAR	The MD5 encryption key
head	VARCHAR	Avatar
register_date	DATETIME	The date of registration
last_login_date	DATETIME	The date of last time login
login_count	INT	The count of login
miaosha_order_id	BIGINT	The id of miaosha order
miaosha_goods_id	BIGINT	ID of miaosha goods

miaosha_order		
miaosha_order_id(PK)	BIGINT	The ID of miaosha_order
user_id	BIGINT	The ID of user
order_id	BIGINT	The ID of order
goods_id	BIGINT	The ID of good
miaosha_user_id(FK)	BIGINT	the ID of miaosha user

miaosha_goods		
miaosha_good_id(PK)	BIGINT	The ID of miaosha_goods

goods_id	BIGINT	The ID of goods
miaosha_price	DECIMAL(10,2)	The price of this bargain
start_date	DATETIME	The start of the discount promotion
end_date	DATETIME	The end of the discount promotion
miaosha_order_id(FK)	BIGINT	The ID of miaosha order
miaosha_order_miaosha_user_id(FK)	BIGINT	The joint ID of order and user

order_info		
id(PK)	BIGINT	The ID of order
user_id	BIGINT	The ID of user
goods_id	BIGINT	The ID of goods
delivery_addr_id	BIGINT	The ID of delivery address
goods_name	VARCHAR	The name of goods
goods_count	INT	The number of goods
goods_price	DECIMAL	The price of goods
order_channel	TINYINT	The platform(Android, IOS, PC)
status	TINYINT	The status
create_date	DATETIME	The time to create
pay_date	DATETIME	The day to pay money
miaosha_goods_id	BIGINT	The id of miaosha goods

● Closing section

We learned a lot doing this project. We did a lot of testing and researching when we were trying to integrate Rabbitmq with sqringboot. Initially we tried to use linux to install them, but our of our computer are not in linux system, we only have some virtual machine on our devices. So we just switched to windows and install using chocolaty. And it was surprising simple. We thought the windows version could be much harder to configure. Another thing was that, when we first use Jmeter to test our performance, we always get 100% error rate. It turns out that our tcpip port was set to 10000 max. And we need to set it to 65534, the maximum data flow it can take in theory. Then all the errors were gone. We were excited because we got stuck on this one for a long time.