Университет ИТМО

Кафедра ВТ

Системы баз данных

Курсовая работа

Выполнил студент гр. P3315

Назарьев Сергей

Санкт-Петербург

2016

1. Предметная область

Всё большую популярность в мире начинают иметь сетевые игры (в том числе MMOFPS, MOBA и т.д.), и эта популярность порождает новый рынок, рынок внутриигровых предметов. Моя задача: помочь этому рынку и поднять экономику игрового сервера SA:MP за счёт небольшого сервиса по анонимной продаже аккаунтов пользователей.

Предполагается, что есть небольшой набор характеристик, которые помогают правильно оценивать стоимость аккаунта: его уровень в игре, сумма внесённых на внутриигровой счёт реальных денег, уровень заполненности инвентаря и т.д.

1. Модель данных

Игровые аккаунты (accounts:%accid%): lvl, cash, donate, nickname, price

Покупки (purchases:%purchid%): account\_id, date

Предметы (items:%accid%): %itemname% => %itemcount%

1. Пример CRUD-функций и 1 зачётное задание

<?php

**require** 'Predis/Autoloader.php';

Predis\Autoloader**::**register();

function switcher(**$text**,**$arrow=**0){

**$str**[0] **=** **array**('й' **=>** 'q', 'ц' **=>** 'w', 'у' **=>** 'e', 'к' **=>** 'r', 'е' **=>** 't', 'н' **=>** 'y', 'г' **=>** 'u', 'ш' **=>** 'i', 'щ' **=>** 'o', 'з' **=>** 'p', 'х' **=>** '[', 'ъ' **=>** ']', 'ф' **=>** 'a', 'ы' **=>** 's', 'в' **=>** 'd', 'а' **=>** 'f', 'п' **=>** 'g', 'р' **=>** 'h', 'о' **=>** 'j', 'л' **=>** 'k', 'д' **=>** 'l', 'ж' **=>** ';', 'э' **=>** '\'', 'я' **=>** 'z', 'ч' **=>** 'x', 'с' **=>** 'c', 'м' **=>** 'v', 'и' **=>** 'b', 'т' **=>** 'n', 'ь' **=>** 'm', 'б' **=>** ',', 'ю' **=>** '.','Й' **=>** 'Q', 'Ц' **=>** 'W', 'У' **=>** 'E', 'К' **=>** 'R', 'Е' **=>** 'T', 'Н' **=>** 'Y', 'Г' **=>** 'U', 'Ш' **=>** 'I', 'Щ' **=>** 'O', 'З' **=>** 'P', 'Х' **=>** '[', 'Ъ' **=>** ']', 'Ф' **=>** 'A', 'Ы' **=>** 'S', 'В' **=>** 'D', 'А' **=>** 'F', 'П' **=>** 'G', 'Р' **=>** 'H', 'О' **=>** 'J', 'Л' **=>** 'K', 'Д' **=>** 'L', 'Ж' **=>** ';', 'Э' **=>** '\'', '?' **=>** 'Z', 'ч' **=>** 'X', 'С' **=>** 'C', 'М' **=>** 'V', 'И' **=>** 'B', 'Т' **=>** 'N', 'Ь' **=>** 'M', 'Б' **=>** ',', 'Ю' **=>** '.',);

**$str**[1] **=** array ( 'q' **=>** 'й', 'w' **=>** 'ц', 'e' **=>** 'у', 'r' **=>** 'к', 't' **=>** 'е', 'y' **=>** 'н', 'u' **=>** 'г', 'i' **=>** 'ш', 'o' **=>** 'щ', 'p' **=>** 'з', '[' **=>** 'х', ']' **=>** 'ъ', 'a' **=>** 'ф', 's' **=>** 'ы', 'd' **=>** 'в', 'f' **=>** 'а', 'g' **=>** 'п', 'h' **=>** 'р', 'j' **=>** 'о', 'k' **=>** 'л', 'l' **=>** 'д', ';' **=>** 'ж', '\'' **=>** 'э', 'z' **=>** 'я', 'x' **=>** 'ч', 'c' **=>** 'с', 'v' **=>** 'м', 'b' **=>** 'и', 'n' **=>** 'т', 'm' **=>** 'ь', ',' **=>** 'б', '.' **=>** 'ю','Q' **=>** 'Й', 'W' **=>** 'Ц', 'E' **=>** 'У', 'R' **=>** 'К', 'T' **=>** 'Е', 'Y' **=>** 'Н', 'U' **=>** 'Г', 'I' **=>** 'Ш', 'O' **=>** 'Щ', 'P' **=>** 'З', '[' **=>** 'Х', ']' **=>** 'Ъ', 'A' **=>** 'Ф', 'S' **=>** 'Ы', 'D' **=>** 'В', 'F' **=>** 'А', 'G' **=>** 'П', 'H' **=>** 'Р', 'J' **=>** 'О', 'K' **=>** 'Л', 'L' **=>** 'Д', ';' **=>** 'Ж', '\'' **=>** 'Э', 'Z' **=>** '?', 'X' **=>** 'ч', 'C' **=>** 'С', 'V' **=>** 'М', 'B' **=>** 'И', 'N' **=>** 'Т', 'M' **=>** 'Ь', ',' **=>** 'Б', '.' **=>** 'Ю', );

**return** **strtr**(**$text**,**isset**( **$str**[**$arrow**] ) ? **$str**[**$arrow**] : **array\_merge**(**$str**[0],**$str**[1]));

}

/\* Абстракция в виде класса нужна только, чтобы

мне иметь общую для всех переменную client \*/

class RedisAccounts {

var **$client**;

function RedisAccounts() {

**$this->**client **=** **new** Predis\Client();

}

/\*\*

\* Добавляет аккаунт в БД

\* **@param** account dictionary с информацией об аккаунте

\* **@return** id только что созданного аккаунта

\*/

function addAccount( **$account** ) {

**$account\_id** **=** **$this->**client**->**incr( "next\_account\_id" );

**$this->**client**->**hmset( "account:**$account\_id**", **$account** );

**return** **$account\_id**;

}

/\*\*

\* Добавляет информацию о предметах аккаунта в БД

\* **@param** account\_id ID аккаунта

\* **@param** items dictionary с предметами

\*/

function addItems( **$account\_id**, **$items** ) {

**$this->**client**->**hmset( "items:**$account\_id**", **$items** );

}

/\*\*

\* Добавляет совершенные покупки аккаунтов в БД

\* **@param** purchase dictionary с информацией о покупке

\* **@return** id только что добавленной в систему покупки

\*/

function addPurchase( **$purchase** ) {

**$purchase\_id** **=** **$this->**client**->**incr( "next\_purchase\_id" );

**$this->**client**->**hmset( "purchase:**$purchase\_id**", **$purchase** );

**return** **$purchase\_id**;

}

/\*\*

\* Добавляет совершенные покупки аккаунтов в БД

\* **@param** purchase\_id id совершенной покупки

\*/

function removePurchase( **$purchase\_id** ) {

**$this->**client**->**del( "purchase:**$purchase\_id**" );

}

/\*\*

\* Получить информацию об аккаунте

\* **@param** account\_id id аккаунта

\* **@return** ассоциативный массив с информацией об аккаунте

\*/

function getAccount( **$account\_id** ) {

**return** **$this->**client**->**hgetall( "account:**$account\_id**" );

}

function getItems( **$account\_id** ) {

**return** **$this->**client**->**hgetall( "items:**$account\_id**" );

}

/\*\*

\* Изменить стоимость аккаунта

\* **@param** account\_id id аккаунта

\* **@param** new\_price новая стоимость аккаунта

\* **@return** ассоциативный массив с информацией об аккаунте

\*/

function changeAccountPrice( **$account\_id**, **$new\_price** ) {

**return** **$this->**client**->**hmset( "account:**$account\_id**",

**array**( "price" **=>** **$new\_price** )

);

}

/\*\*

\* Сгенерировать инстанс покупки аккаунта

\* **@param** account\_id id аккаунта

\* **@return** ассоциативный массив с информацией о покупке

\*/

function generatePurchase( **$account\_id** ) {

**return** **array**( "account" **=>** **$account\_id**,

"date" **=>** **date**("Y-m-d H:i:s", **rand**( 0, 9999999)) );

}

function findStringInAccounts(**$text**, **$last**) {

**$accounts** **=** **array**();

**$sw** **=** switcher( **$text** );

**for**( **$i** **=** 1; **$i** **<=** **$last**; **$i++**) {

**$account** **=** **$this->**getAccount(**$i**);

**if**( **strstr**(**$account**["nickname"], **$text** ) **!==** FALSE **||**

**strstr**(**$account**["nickname"], **$sw** ) **!==** FALSE )

**$accounts**[] **=** **$account**;

}

**return** **$accounts**;

}

function generateItems() {

**$listItems** **=** **array**( "Knife", "AK-47", "Bulbulator", "Scissors", "Rifle", "M4-A1" );

**shuffle**(**$listItems**);

**$listItems** **=** **array\_slice**( **$listItems**, 0, 3);

**$items** **=** **array**();

**$itemMin** **=** 1;

**$itemMax** **=** 20;

**foreach**( **$listItems** **as** **$value** )

**$items**[**$value**] **=** **rand**( **$itemMin**, **$itemMax** );

**return** **$items**;

}

/\*\*

\* Сгенерировать аккаунт

\* **@param** account\_id id аккаунта

\* **@return** ассоциативный массив с информацией об аккаунте

\*/

function generateAccount() {

**$lvlMin** **=** 10;

**$lvlMax** **=** 100;

**$cashMin** **=** 0;

**$cashMax** **=** 1000000000;

**$donateMin** **=** 0;

**$donateMax** **=** 1000000000;

**$nicknames** **=** **array**( "Grigoriy", "Vladimir", "Sergey", "Elena", "Nagibator", "Aberman" );

**$yearMin** **=** 1996;

**$yearMax** **=** 2042;

**$priceMin** **=** 50;

**$priceMax** **=** 15000;

**return** **array**( "lvl" **=>** **rand**( **$lvlMin**, **$lvlMax** ),

"cash" **=>** **rand**( **$cashMin**, **$cashMax** ),

"price" **=>** **rand**( **$priceMin**, **$priceMax** ),

"donate" **=>** **rand**( **$donateMin**, **$donateMax** ),

"nickname" **=>** **$nicknames**[**rand**( 0, **count**( **$nicknames** )**-**1 )] **.** **rand**(**$yearMin**, **$yearMax**)

);

}

}

**$redis** **=** **new** RedisAccounts();

**$count** **=** 100000;

**$lastAccountId** **=** **$count**;

**$lastPurchaseId** **=** 0;

/\*\*

\* Генерируем стержневую и характеристичные сущности

\*/

**for** (**$i** **=** 1; **$i** **<=** **$count**; **$i++**) {

**$lastAccountId** **=** **$redis->**addAccount( **$redis->**generateAccount( ) );

**$lastPurchaseId** **=** **$redis->**addPurchase( **$redis->**generatePurchase( **$lastAccountId** ) );

**$redis->**addItems( **$lastAccountId**, **$redis->**generateItems() );

}

/\*\*

\* Делаем выборку стержневых сущностей с агрегацией характеристичных сущностей

\*/

**for** (**$i** **=** 1; **$i** **<=** **$count**; **$i++**) {

**$account** **=** **$redis->**getAccount(**$i**);

**$account**["items"] **=** **$redis->**getItems(**$i**);

**print\_r**( **$account** );

}

/\*\*

\* Производим поиск в имени аккаунтов всех сущностей, с учётом возможной смены регистра (см. функцию **findStringInAccounts**)

\*/

**print\_r**(**$redis->**findStringInAccounts( "Gri", **$count**));

?>