**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ “КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”**

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВИЙ ПРОЕКТ**

з дисципліни “Бази даних”

напряму підготовки 6.050103 – Програмна інженерія

на тему:

(назва теми)

|  |  |  |
| --- | --- | --- |
| **Студент**  **групи** КП-51 | (ПІБ) | (підпис) |
| **Викладач**  **к.т.н, доцент кафедри СПіСКС** | **Петрашенко А.В.** | (підпис) |

Захищено з оцінкою

Київ – 2018

АНОТАЦІЯ

Дана курсова робота присвячена створенню додатку для аналізу успішності спортсменів.

Додаток являє собою консольну програму, яка містить деякі функції для виконання описаного аналізу.

По-перше, це функції формування набору даних за допомогою наданого компанією **© “World Archery”** **(далі – WA)** **API (далі – WA API)** і занесення цих сутностей до бази даних.

По-друге, це функції для аналізу різноманітної зібраної інформації про спортсменів та їх досягнень у змаганнях, пошук певних залежностей та формування аналітичних результатів серед зібраних даних.

Таким чином, даний додаток може буде застосований аналітиками або просто зацікавленими у результатах спортивних змагань особами для знаходження цікавих закономірностей, інформація про які може бути оброблена та у подальшому застосована.

ЗМІСТ

[АНОТАЦІЯ 3](#_Toc516443847)

[ЗМІСТ 4](#_Toc516443848)

[ВСТУП 5](#_Toc516443849)

[1. АНАЛІЗ ІНСТРУМЕНТАРІЮ ДЛЯ ВИКОНАННЯ КУРСОВОГО ПРОЕКТУ 7](#_Toc516443850)

[2. СТРУКТУРА БАЗИ ДАНИХ 9](#_Toc516443851)

[3. ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 10](#_Toc516443852)

[*3.1.* *Загальна структура програмного забезпечення* 10](#_Toc516443853)

[*3.2.* *Опис модулів програмного забезпечення* 10](#_Toc516443854)

[*3.3.* *Опис основних алгоритмів роботи* 12](#_Toc516443855)

[4. АНАЛІЗ ФУНКЦІОНУВАННЯ ЗАСОБІВ МАСШТАБУВАННЯ 14](#_Toc516443856)

[5. ОПИС РЕЗУЛЬТАТІВ АНАЛІЗУ ПРЕДМЕТНОЇ ГАЛУЗІ 16](#_Toc516443857)

[ВИСНОВКИ 17](#_Toc516443858)

[6. ЛІТЕРАТУРА 19](#_Toc516443859)

[7. ДОДАТКИ 20](#_Toc516443860)

[7.1. ГРАФІЧНІ МАТЕРІАЛИ 20](#_Toc516443861)

[7.2. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ 23](#_Toc516443862)

ВСТУП

Розроблений проект у рамках цієї курсової роботи – це інструмент, за допомогою якого можна збирати дані з **WA API** та проводити аналіз зібраних результатів.

Галуззю застосування розробки та її функціонал в стислому варіанті може описаний як збір, фільтрація та аналіз результатів різних вікових категорій та рівня підготовки спортсменів, які обрали в якості спорту дисципліну стрільби з олімпійського виду лука, з метою прогнозування та вдосконалення спортивного процесу.

Слід зауважити, що обрана спортивна дисципліна є винайденою досить давно, і, хоч вона не є досить популярною у порівнянні з іншими, кожного року у багатьох країнах є багато спортсменів, серед яких проводиться безліч змагань, і кожні змагання тягнуть за собою ретельне відслідковування результатів усіх його етапів. І хоч не всі змагання проводяться на світовому рівні, і не всі результати вносяться у електрону систему, компанія **WA** проводить величезний об'єм робіт, і намагається зберігати у базах даних у електроному виді якомога більше результатів. Окрім цього, дисципліна стрільби з лука є досить варіативним видом спорту, і серед неї можна відокремити дуже багато її видів, кожний з яких можна спростити до того, що це є стрільба у мішень з метою влучення зони, яка принесе спортсмену найбільшу кількість балів, але водночас способи проведення змагань та відмінності у дозволеного для певного виду змагань інструментарію спортсмена призводить до того, що дуже складно сконструювати систему, яка б могла б однаково ефективно працювати з такими різними наборами даних.

Отже, з метою проведення більш детального аналізу, об’єм даних був обмежений, а саме таким чином, що в даній “початковій” версії застосунку проводиться аналіз результатів спортсменів, які користуються тільки олімпійським видом лука, і були взяті набори результатів з п’яти змагань, а саме: Олімпійських Ігор з 2000 по 2016 роки (2000, 2004, 2008, 2012 та 2016 відповідно).

Окрім реалізованого у рамках даної роботи функціоналу та змоги користуватися ним, безпосередньою метою реалізації якого і був даний проект, під час розроблення було також вдосконалено навички роботи з обраною мовою програмування, тобто Python, у вигляді вивчення відповідних нових бібліотек, та покращено навички роботи з обраною СКБД. Результатом вищеописаного і стало створення програмного застосунку, який дозволяє проводити аналіз даних спортивних змагань.

1. АНАЛІЗ ІНСТРУМЕНТАРІЮ ДЛЯ ВИКОНАННЯ КУРСОВОГО ПРОЕКТУ

* *Обрана СКБД:* ***MongoDB***.

Для даного програмного застосунку була вибрана СКБД MongoDB. MongoDB – це документно-орієнтована СКБД, класифікована як NoSQL, яка не потребує опису схеми таблиць.

NoSQL підхід апріорі не є настільки ж валідованим, як SQL-підхід, властивий таким БД як PostgreSQL, MySQL або MS SQL, і це призводить до того, що, працюючи з базами даних типу MongoDB або Redis, розробник має сам реалізовувати перевірки на валідність відправлених та прийнятих на/з бази даних набори інформації. Але, водночас, це дозволяє використовувати одну й ту саму базу даних (або окрему колекцію) для зберігання різних наборів даних, які за логікою класифікуються як такі, які треба зберігати поряд. Якщо в них дещо відрізняється схема, було б неможливо зберігати їх в межах однієї таблиці, але для NoSQL підходу з колекціями це не є завадою.

Окрім цього, сучасні системи, розширення яких і в плані кількості даних, і в плані різних наборів даних відбувається дуже часто, на базі SQL підходу робити дуже важко, адже додання певної нової сутності може потягти за собою зміну в інших, що не завжди є легко можливим. У NoSQL базах даних додання нової сутності, навіть якщо вона певні логічні зв’язки з іншими сутностями, є доволі легким процесом, адже вже створені сутності без проблем дозволяють редагувати їх структуру, додаючи, або, навпаки, прибираючи наведені у них поля для збереження даних.

Також, даній конкретній СКБД властивий ряд визначних можливостей, серед яких:

* *Файлове сховище:* MongoDB може бути використана у якості файлового сховища з балансуванням навантаження та реплікацією даних.
* *Реплікація даних та балансування навантаження:* MongoDB може працювати з набором реплік. Такий набір складається з, теоретично, двох (а практично – мінімум трьох) або більше копій даних. Кожний з екземплярів набору реплік може в будь-який момент виступати у ролі основної (PRIMARY) або допоміжної (SECONDARY) репліки. Усі операції читання та запису за замовченням виконуються з основною реплікою. Допоміжні репліки підтримують в актуальному стані копії даних. У випадку, коли основна репліка дає збій, набір реплік вибирає, яка з реплік має стати основною.

*Масштабування:* MongoDB масштабується горизонтально, використовуючи шардінг. Користувач обирає ключ шарда (певне поле, наявне в усіх сутностях), який визначає, як будуть розподілені дані у колекції. Дані розподіляються на діапазони (в залежності від ключа шарда) та розподіляються по шардам.

* *Обрана бібліотека для аналізу даних:* ***Pandas*** *(основна):*

Під час розробки даного програмного застосунку, постала проблема з роботою на різних наборах даних (сутностях), які, у свою чергу, мають прямі та вкладені залежності з іншими сутностями. Окрім цього, такі дані були представлені не одиничними сутностями, а списками або масивами. Для спрощення роботи з ними та створення більш зрозумілого коду була обрана вищезгадана бібліотека (яка, у свою чергу, збудована на базі іншої, NumPy, яка вважається бібліотекою нижчого рівня). Ключовою можливістю *Pandas* є саме можливість описання багатовимірних структурованих наборів даних.

Окрім цього, *Pandas* також дозволяє й обмін даними між цими структурами у пам’яті; наявний вбудований інструментарій для об’єднання даних та способи обробки відсутньої інформації (що при виборі роботи за NoSQL підходом є досить вагомою властивістю); переформатування наборів даних (в т. ч. створення сусідніх таблиць за потреби), розширені можливості індексування, вибірка з великих наборів даних (у даному проекті робота відбувається здебільшого об’ємними наборами даних), злиття та об’єднання наборів даних та багато інших.

Окрім (або замість) основної бібліотеки, також могли бути застосовані такі бібліотеки, як *NumPy*, *SciPy*, *Matplotlib*. Слід зауважити, що, так чи інакше, усі наведені бібліотеки пов’язані між собою. Таким чином, вони або включають в себе аналогічні можливості з інших бібліотек, або можуть конвертувати подання своїх даних на виході для використання іншими бібліотеками, або навіть напряму містять функції, використання яких потребує встановлення додаткових бібліотек зі списку.

1. СТРУКТУРА БАЗИ ДАНИХ

Структура бази даних наведена у ДОДАТКУ А (рис. 1).

Слід зауважити, що у MongoDB як у базі даних NoSQL типу відсутнє пряме поняття терміну FK (Foreign Key), але MongoDB дозволяє пов’язувати сутності (документи) між собою за рахунок додання поля сутності ObjectId, яка буде містити значення \_id документу з будь-якої іншої (або, за потреби, цієї ж самої) колекції. Таким чином, поля, позначені як поля типу PK, слід інтерпретувати як поля, які містять значення типу ObjectId, і посилаються на сутність з певної іншої колекції (згідно наведеного зв’язку).

1. ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
   1. *Загальна структура програмного забезпечення*

Структура даного програмного застосунку диктується функціоналом, який ця програма надає. Таким чином, в загальному вигляді, структура складається з двох основних частин. Перша основна структурна частина, яку слід використовувати час від часу (для оновлення існуючих у базі даних або додання), - це частина, яка відповідає за збір даних. Так як за джерело для отримання достовірних у межах області програмного застосунку даних вибране **WA API**, ця структура частина складається з модулів (класів, функцій), які відповідають за безпосередню взаємодію з цим REST API, відбір необхідних даних, приведення їх у придатний для даного програмного застосунку вигляд та запис наповнення за допомогою бібліотеки *PyMongo* відповідної бази даних.

Другою (основною) структурною частиною є безпосередньо набір модулів (класів, функцій) для аналізу занесених (та, відповідно, отриманих) у попередньому пункті даних, та пошук залежностей; максимальних, мінімальних або середніх значень, фільтрування результатів для окремого спортсмену, або отримання результатів для спортсменів з певної країни тощо. У реалізації даної частини широко використовується бібліотека *Pandas*, описана раніше.

* 1. *Опис модулів програмного забезпечення*

Таким чином, програма складається з 4 основних модулів.

* Модуль зв’язку з **WA API**:

Представлений класом WA\_API. Клас представляє собою набір функцій, у середині яких лежить відправлення HTML запитів на ресурс **WA** з різними наборами параметрів URL строки для отримання відповідних наборів даних: інформації про спортсменів, змагання, результати спортсменів у межах даних змагань або детальної інформації результатів спортсменів (результат кожного з влучень).

Також, в цьому класі можна виділити підмодуль, який складається з функцій, назва яких починається з “db\_\_”. Дані функції, які частково базуються на інших функціях з даного класу, отримують та підготовлюють дані таким чином, що після повернення даних цими функціями ці набори даних мають правильну для бази даних структуру і можуть бути одразу ж бути передані в модуль керування базою даних для запису.

* Модуль роботи з базою даних:

Представлений класом MongoManage. Клас представляє собою набір функцій для роботи з базою даних. Фактично є обгорткою над бібліотекою PyMongo для конкретної бази даних, представленої у цьому проекті.

Клас створений для того, щоб “сховати” більш абстрактну бібліотеку, створену для взаємодії з MongoDB в цілому, і дозволити конкретизувати запити для работи з безпосередньою базою даних проекту. Але усі функції, відповідно, так чи інакше використовують функціонал з вищезгаданої бібліотеки для роботи з MongoDB в мові Python.

* Модуль зв’язку модулів для роботи з **WA API** та роботи з базою даних:

Представлений класом FillDB. Це – невеликий клас, який є обгорткою над двома описаними вище класами. Сам клас не містить логіки як такової, а просто у коректній послідовності викликає надані класами WA\_API та MongoManage функції, для того, щоб, першим кроком, отримати ці дані ззовні, і, другим кроком, занести отримані дані у базу даних.

* Модуль взаємодії з користувачем та аналізу даних:

Не дивлячись на те, що це два модуля з абсолютно різними задачами, через те, що ці два модуля у рамках даного програмного застосунку є досить невеликими, було прийняте рішення реалізувати функції обох у межах однієї модульної частини

Модуль представлений класом Menu. З точки зору взаємодії з користувачем, цей клас керує відображенням безпосередньо самого меню, реагує на запити від користувача та виводить результати аналізу.

При цьому, для кожного пункта меню описана своя функція реакції на вибір того чи іншого пункта. При цьому, дані функції містять логіку аналізу у середині, а не користуються модулем ззовні для отримання результатів.

* 1. *Опис основних алгоритмів роботи*

Опис основних алгоритмів роботи буде приведений на прикладі послідовного використання функцій програми для отримання кінцевого результату (виведення результатів аналізу), з детальним розбором кожного з використовуваних модулів.

Отже, першим етапом є збір даних з джерела ззовні для внесення їх у створену базу даних.

Використовуючи клас та функції з нього WA\_API, ми отримуємо дані. Дані отримуються шляхом відправлення HTML запиту на REST API **WA**, звідки приходить відповідь у JSON форматі. Інформація з цієї відповіді приводиться до зручного для роботи типу словника, звідки видаляється уся непотрібна у рамках даного програмного застосунку інформація, а інша інформація аналізується та конвертується у такий тип, який визначений придатним для спроектованих у рамках бази даних колекцій.

Другим етапом є внесення зібраних даних у базу даних.

Отже, після того, як інформація була перетворена на тип, готовий до занесення у базу даних, ми використовуємо клас MongoManage для внесення цієх інформації у нашу локальну базу даних. У середині цього класу створюється т. наз. “клієнт” (бібліотека *PyMongo*), у який передаються такі налаштування підключення до бази даних, як хост, порт та Replica Set’и, і через функції, надані цим клієнтом у рамках бібліотеки, відбувається внесення наборів даних у відповідні колекції.

Для спрощення виконання перших двох етапів та зменшення можливості для помилки під час перенесення даних з віддаленого ресурсу у локальну базу даних, використовується модуль FillDB (у рамках класу FillDB), описаний раніше, який поетапно, для кожної з спроектованих колекцій, виконує збір інформації та її внесення у базу даних.

Третім та четвертим пунктами є взаємодія з користувачем та аналіз даних.

Дані пункти реалізовані у рамках класу Menu. Для користувача виводиться меню, у рамках якого він може вибрати певний пункт. За вивід меню та реакції на вибір користувача відповідає функція init\_menu().

Після обрання користувачем певного пункту меню, у якості реакції на вибір буде визиватися одна з інших функцій, наведених у даному класі. Усі можливі для вибору пункти меню, окрім пунктів виводу допомоги та пунктів завершення програми, мають під собою проведення аналізу, та вивід його результатів. Отже, якщо обраний пункт меню, який передбачає аналіз, цей аналіз буде проведено у середині відповідної функції. Після проведення, результати аналізу буде виведено у рамках цієї ж самої функції, і програма знов повернеться до функції init\_menu() для очікування від користувача обрання чергового пункту меню. Програма завершає свою роботу після обрання користувачем пункту вихода у меню.

1. АНАЛІЗ ФУНКЦІОНУВАННЯ ЗАСОБІВ МАСШТАБУВАННЯ

В даному розділі будуть описані засоби масштабування та реплікації даних у рамках бази даних, використовуваної у цьому програмному застосунку.

Як і було зазначено раніше, під час проведення збору даних були зібрані результати всього з п’яти змагань (з п’яти Олімпійських Ігор). Отже, не дивлячись на те, що з інформацією про самі змагання слідує й супутня інформація про спортсменів, які брали участь у них, й результати цих спортсменів, об’єм даних виявився замалим для застосування можливостей MongoDB у плані горизонтального масштабування. Таким чином, весь набір даних, з яким відбувається взаємодія у рамках даного застосунку, знаходиться на одному комп’ютері (одному сервері), а результати швидкодії щодо доступу до цього набору даних показують те, що масштабування не є затребуваним при таких розмірах збережених даних.

Реплікація ж даних, яка забезпечує створення певного набору копій даних з метою їх резервування, має сенс бути застосованою для довільного розміру набору даних і рекомендується для застосування у базах даних будь-яких масштабів, якщо вони використовуються у т. наз. Production’і. Таким чином, у рамках даної роботи була проведена реплікація, і було створено три окремих бази дани, об’єднані в єдиний Replica Set. Ці бази даних функціонують в один і той самий час. При цьому, одна з баз даних має статус PRIMARY.

У рамках створеного Replica Set’у була створена одна база даних зі статусом PRIMARY, одна – зі статусом SECONDARY та одна – зі статусом ARBITER. База даних зі статусом ARBITER не містить ніяких даних у собі, і її роль у рамках Replica Set’у полягає в тому, щоби вирішити, яка з інших наявних *працюючих* *на даний момент* баз даних має функціонувати зі статусом PRIMARY. Після вибору PRIMARY бази даних, усі інші наявні бази даних будуть примусово функціонувати у статусі SECONDARY.

Усі операції запису, направлені на даний Replica Set, застосовуються саме до бази даних зі статусом PRIMARY, а інші бази даних, після проведення відповідної операції запису на PRIMARY базу даних, копіюють “поведінку” цих операцій на свій набір даних.

Операції читання, при цьому, за замовчуванням застосовуються до PRIMARY бази даних, але існують й інші режими налаштування операції читання. Таким чином, для даного програмного застосунку був обраний режим “PrimaryPreferred”, який означає, що операції читання будуть намагатися проводиться на базі даних зі статусом PRIMARY, але, якщо вона є недоступною (але при цьому ще не була обрана нова база даних для роботи у статусі PRIMARY), операція читання буде проводитися на якусь базу зі статусом SECONDARY.

Отже, у рамках даного застосунку був спроектований та протестований описаний вище Replica Set. Тестування відбувалося у спосіб проведення запису на PRIMARY базу даних та слідкування за тим, щоб відповідна операція була застосована і до SECONDARY бази даних, а також в спосіб відключення PRIMARY бази даних та слідкування за тим, щоб наявна SECONDARY база даних змінила свій статус на PRIMARY та зв’язок застосунку із базою даних не обірвався.

1. ОПИС РЕЗУЛЬТАТІВ АНАЛІЗУ ПРЕДМЕТНОЇ ГАЛУЗІ

проаналізувати результати виконання алгоритмів, що були використані в роботі. Дати текстовий опис, а результати у графічні формі надати у додатку.

У даній курсовій роботі мають змогу проводитися наступні види аналізу на заданому наборі даних:

* Детальний аналіз певного змагання (аналіз окремих зроблених пострілів)

Даний пункт передбачає вказання певного змагання для отримання результатів усіх пострілів, зроблених в межах цього змагання, та підрахунку процентного співвідношення кількості пострілів, які призвели до отримання певної кількості очок, до кількості усіх зроблених пострілів.

* Аналіз середніх результатів усіх змагань

Даний пункт передбачає аналіз (проведений окремо для кожного зі змагань) з метою отримання середніх для даного змагання результатів кваліфікаційних раундів (окремо для чоловічої та жіночої категорій)

* Аналіз максимальних результатів у межах усіх змагань

Даний пункт передбачає аналіз (проведений серед усіх змагань) з метою визначення максимальних результату у межах кваліфікаційного раунду (окремо для чоловічої та жіночої категорій)

* Аналіз досягнень одного спортсмена у межах усіх змагань

Даний пункт передбачає вказання певного спортсмена для отримання результатів кваліфікаційних раундів цього спортсмена у межах усіх змагань, в яких він брав участь (індивідуально чи у складі команди), та порівняння цих результатів.

* Аналіз досягнень усіх спортсменів певної країни у межах усіх змагань

Даний пункт передбачає вказання певної країни для отримання результатів кваліфікаційних раундів по всім змаганням усіх спортсменів, які брали участь в даних змаганнях (індивідуально чи у складі команди), та порівнянні цих результатів.

# ВИСНОВКИ

Отже, виконання поставлених задач у рамках даної курсової роботи призвело до розробки програмного застосунку, який, хоч і в невеликих масштабах, але реалізовує описані в ньому функціональні вимоги. Користувач цього програмного застосунку має змогу провести збір даних з певного джерела, зберегти їх у базу даних, та, найважливіше, провести певні види аналізу, розроблені у поточній версій програмного застосунку. Таким чином, мета даного курсового проекту, яка полягала в тому, щоб вдосконалити навички роботи з мовою програмування *Python*, відповідною бібліотекою *Pandas* та СКБД (*MongoDB*, бібліотека *PyMongo*) та створенні програмного застосунку, який дозволив би проводити аналіз даних спортивних змагань, виконана в повному обсязі.

Також, вище вказувалося, що з метою можливості проведення більш детального аналізу (на наборах даних однакового формату) даний програмний застосунок та наведені у ньому аналізи застосовувалися для невеликого набору даних з певної галузі. Але структура даного програмного забезпечення спроектована таким чином, що розширення (вдосконалення) вже наведеного функціоналу та додання нового є досить простим та не призведе до перероблення архітектури програмного застосунку. Таким чином, за бажаннями, розробку даної курсової роботи можна продовжувати, та додавати в неї нові елементи, і, таким чином, продовжувати дослідження у цій області, створюючи більш привабливий для користувачів програмний застосунок, який, при належному рівні розробки, може перетворитися з проекту, створеного у навчальних цілях, на повноцінний застосунок, у використанні якого будуть зацікавлені провідні компанії, так чи інакше пов’язані зі спортивним сегментом дисципліни стрільби з лука, такі як *WIN&WIN*, *Hoyt*, *World Archery* та інші.

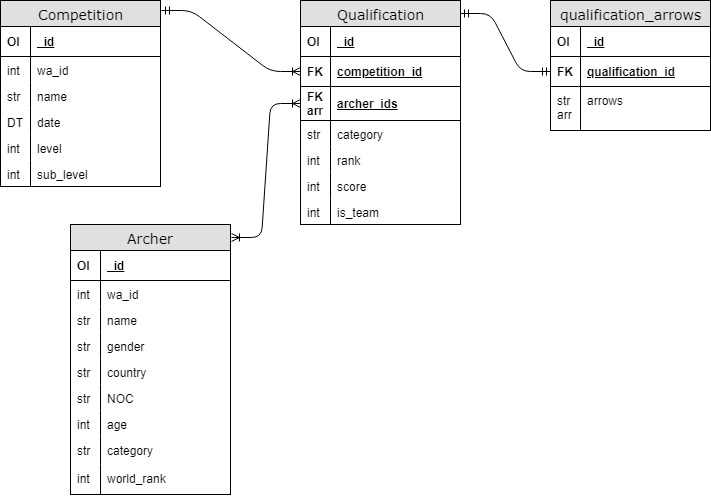
Окремий абзац хотілося б виділити для подяки компанії **World Archery**, яка є найбільшою компанією, яка займається керівництвом усіх крупних змагань та пов’язаних зі стрільбою з лука заходів світового масштабу, представник якої, **Chris Wells**, надав доступ до REST API, за допомогою якого можна було отримати детальні результати усіх змагань світового рівня, які проводилися з 1972 року (такі як Олімпійські Ігри 1972 року) та Кубків світу й Чемпіонатів світу, які проводилися з 1984 року. Без надання доступу до даного ресурсу було б неможливо провести аналіз, який було проведено у рамках даної курсової роботи. Не дивлячись на те, що сам аналіз можна було б застосувати до будь-якого, навіть випадково згенерованого набору даних, це потребувало б реалізації додаткового модулю для генерації цих даних, а аналіз, в свою чергу, не зміг би відобразити реальний, наявний на даний момент стан речей у спорті.

# ЛІТЕРАТУРА

* + - 1. Сулема, Є.С. Посібник з дипломного проектування за напрямами підготовки “Прикладна математика”, “Комп’ютерна інженерія”, “Програма інженерія” [Текст] / Сулема, Є.С, О.А. Молчанов, В.П. Тарасенко, І.А. Жуков [та ін.]; під заг. ред. І.А. Дички. — К. : НТУУ «КПІ», 2011. — 224 с.
      2. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95 [Текст]. — К. : Держстандарт України, 1998.
      3. 10 Minutes to Pandas [Електронний ресурс]. — Режим доступу: <https://pandas.pydata.org/pandas-docs/stable/10min.html>
      4. Яргер, Р. NoSQL. Базы данных для небольших предприятий и Интернета [Текст] / Р. Яргер, Дж. Риз, Т. Кинг. — СПб. : Символ- Плюс, 2000. — 560 с.

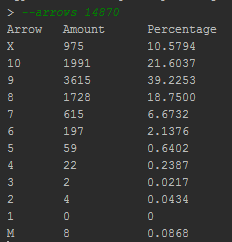
# ДОДАТКИ

## ГРАФІЧНІ МАТЕРІАЛИ



*Рис. 1. Структура бази даних.*

Далі будуть наведені скріншоти консольного вікна з результатами аналізів, які можна провести у рамках даного програмного застосунку.



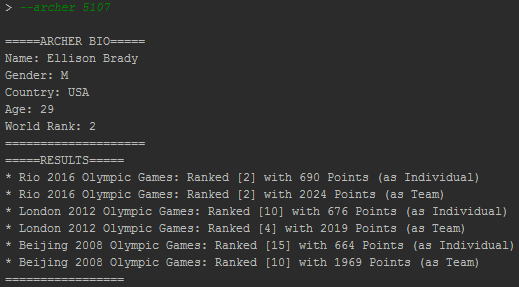
*Рис. 2. Детальний аналіз певного змагання.*



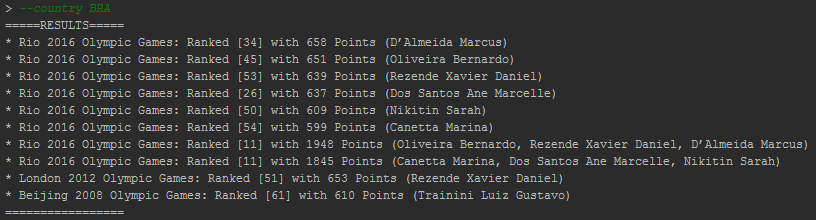
*Рис. 3. Аналіз середніх результатів усіх змагань.*

**

*Рис. 4. Аналіз максимальних результатів у межах усіх змагань.*

**

*Рис. 5. Аналіз досягнень одного спортсмена у межах усіх змагань.*

**

*Рис. 6. Аналіз досягнень усіх спортсменів певної країни у межах усіх змагань.*

## ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

Повний код курсової роботи наведений за даним посиланням:

<https://github.com/Tayum/di0d/tree/master/courses/database_discipline/course3_term2/coursework/>

*(GitHub репозиторій)*

|  |
| --- |
| **Class WA\_API (функція проведення запиту до ресурсу з REST API)** |
| def \_\_get(self, querystring, as\_json=True, info="items"):  """  :param querystring:  querystring is a dictionary with URL parameters  :param as\_json:  True - Use json.loads() to create a dictionary from the response from API  False - return response and plain HTML  :param info:  "items" - will return only items from response  "pageInfo" - will return only pageInfo from response  (others) - return both  """  # add the default parameter to query  querystring['v'] = "3"  response = requests.request("GET", self.base\_url, params=querystring, verify=False)  if not as\_json:  return response.text  js = response.text  js = json.loads(js)  if info == "items":  return js['items']  elif info == "pageInfo":  return js['pageInfo']  else:  return js |

|  |
| --- |
| **Class MongoManage (функція додання нової сутності у певну колекцію в базі даних)** |
| def insert(self, collection, obj):  """  Insert for the collections with no dependencies with other collections  OR where dependencies has already been resolved  """  try:  result = self.db[collection].insert\_one(obj)  return result.inserted\_id  except:  print("{0} collection: failed to insert object: {1}".format(collection, obj))  return -1 |

|  |
| --- |
| **Class Menu (основна функція меню)** |
| def init\_menu(self):  print("Type -h or --help for printing help")  while True:  choice = input("> ").strip()  try:  if choice == "-q" or choice == "--quit":  break  elif choice == "-h" or choice == "--help":  self.\_\_print\_help()  elif "--arrows" in choice:  try:  competition\_id = int(choice.replace("--arrows", "").strip())  except ValueError:  competition\_id = COMPETITION\_ID  self.\_\_arrows\_percentage(competition\_id)  elif "--mean" == choice:  self.\_\_mean\_results()  elif "--max" == choice:  self.\_\_max\_result()  elif "--archer" in choice:  try:  archer\_id = int(choice.replace("--archer", "").strip())  except ValueError:  archer\_id = ARCHER\_ID  self.\_\_archer\_results(archer\_id)  elif "--country" in choice:  country\_noc = choice.replace("--country", "").strip()  if not country\_noc:  country\_noc = COUNTRY\_NOC  self.\_\_country\_results(country\_noc)  except:  print("Something went wrong while processing your request.") |

|  |
| --- |
| **Class Menu (одна з функцій проведення аналізу)** |
| def \_\_mean\_results():  competitions = mm.get\_competitions()  for c in competitions:  print("\n====={0}=====".format(c['name']))  qualifications = mm.get\_individual\_qualification\_scores\_within\_competition(c['wa\_id'])  df = pd.DataFrame(qualifications)  print("Men AVG: " + str(df['score'][df['category'] == 'RM'].mean()))  print("Women AVG: " + str(df['score'][df['category'] == 'RW'].mean())) |