DOI: https://doi.org/10.36910/6775-2524-0560-2025-61-11
УДК 004.75
Кабак Леонід Віталійович[1], к.т.н., доцент
http://orcid.org/0000-0001-6267-1772
Мороз Дмитро Максимович[1], Ph.D.
https://orcid.org/0000-0003-2577-3352
Мороз Борис Іванович[1], д.т.н., професор
https://orcid.org/0000-0002-5625-0864
Варех Нонна В'ячеславівна[2], к.н. соц. ком.
https://orcid.org/0000-0002-2779-9225
[1]Національний технічний університет «Дніпровська політехніка», м. Дніпро, Україна.
[2]Університет «Метінвест політехніка», м. Запоріжжя, Україна.

АНАЛІЗ ПРОДУКТИВНОСТІ РОБОТИ БАЗ ДАНИХ РІЗНОГО ТИПУ ТА РІЗНОЇ АРХІТЕКТУРИ

Кабак Л. В., Мороз Д. М., Мороз Б. І., Варех Н. В. Аналіз продуктивності роботи баз даних різного типу та різної архітектури. Метою роботи є аналіз особливостей зберігання та обробки даних та дослідження продуктивності роботи в сучасних інформаційних системах. Методологія вирішення поставленого завдання полягає в проведенні порівняльного аналізу архітектури та показників продуктивності різних конфігурацій системи, які використовуються для збереження та обробки даних, які використовують технологію Big Data, Cloud, Grid. У роботі проведено аналіз архітектури та проведено дослідження швидкодії сучасних систем збереження та обробки даних в залежності від різної архітектури та різних типів баз даних. У результаті проведеного дослідження було доведено на прикладі реляційних баз даних з різними конфігураціями на прикладі СУБД Oracle та стовпчико-орієнтованою базою даних Hbase, що не зважаючи на недоліки стовпчико-орієнтованої бази даних швидкість доступу до даних в неї швидчий за рахунок її архітектури. Однак за рахунок того що стовпчико-орієнтовані бази даних мають значні недоліки а саме вони не підтримують обробку транзакцій та обмежень цілісності сфера їх застосування дуже обмежена. Результати даної роботи можуть бути використані для реалізації ефективної системи Hadoop Big Data, Grid, Cloud.
Ключові слова: Big Data, Hadoop,Grid, Cloud, Hbase, Oracle.

Kabak L., Moroz D., Moroz B., Varekh N. Performance analysis of databases of different types and architectures. The purpose of this work is to analyze the features of data storage and processing and to study the performance of modern information systems. The methodology for solving the stated task involves conducting a comparative analysis of the architecture and performance indicators of various system configurations used for data storage and processing that utilize Big Data, Cloud, and Grid technologies. The paper analyzes the architecture and examines the performance of modern data storage and processing systems depending on different architectures and types of databases. As a result of the conducted research, it was demonstrated—using relational databases with various configurations such as the Oracle DBMS and the column-oriented database HBase—that despite the shortcomings of column-oriented databases, their data access speed is higher due to their architecture. However, since column-oriented databases have significant disadvantages, namely, the lack of support for transaction processing and integrity constraints, their scope of application is quite limited. The results of this work can be used for the implementation of an efficient Hadoop Big Data, Grid, or Cloud system.
Keywords: Big Data, Hadoop,Grid, Cloud, Hbase, Oracle.

Постановка наукової проблеми. В дійсний час в сучасних інформаційних системах використовуються різноманітні технології оброки та збереження даних. Найбільш поширені технології це Big Data, Grid, Cloud database. Ці технології використовують коли виникає необхідність в обробці великих масивів даних. Дослідження технологій збереження та обробки даних, їхня класифікація та вироблення рекомендації щодо використання тієї чи іншої технології в залежності від особливостей та задач, що постають перед розробниками сучасних розподілених інформаційних систем, на цей час є актуальною проблемою.

При використанні технології Big Data ми маємо декілька баз даних які вже вбудовані в інформаційну систему Hadoop, однак найбільш цікаво розглянути саме базу даних HBase, яка є стовпцово-орієнтованою розподіленою базою даних та провести порівняльний аналіз швидкодії збереження, читання даних з відомими технологіями як Orcle Grid та Oracle Cloud.

Отже, основна мета цієї роботи є дослідження технологій збереження даних в розподілених системах та надати рекомендації що до вибору системи збереження даних в залежності від типу завдань.

Аналіз досліджень. В дійсний час хмарні бази даних отримали високе розповсюдження завдяки їх високої доступності та низької вартості. Як правило такі бази даних є розподіленими, тобто вони зберігають данні на декількох серверах у хмарному сховищі. Багато вчених займаються дослідженням продуктивності сучасних хмарних баз даних та розподілених систем. Так у статті [1]

було представлено основну теорію розподілених систем, включаючи визначення, характеристики та обчислювальні моделі, а потім досліджено архітектуру проектування розподілених систем, зосереджуючись на аналізі різних архітектурних шаблонів та протоколів. Висновками цієї роботи є те що розумна архітектура та стратегія оптимізації є ключем до досягнення ефективної розподіленої системи.

У роботі [2] було досліджено технологію SwarmLinda та Anti-Over-Clustering, які також використовуються в розподілених базах даних. У роботі показано що ці технології покращують масштабованість але погіршують продуктивність.

У роботі [3] досліджується роль моделювання даних у часі обробки та обсязі даних розподіленого сховища даних. В роботі досліджені наступні розподілені бази даних: Singlestore, Amazon Redshift та MariaDB Columnstore у двох різних сценаріях доступності пам'яті було використано тест Star Schema Benchmark. У результаті досліджень, які наведені в цій науковій праці, було показано що денормалізація даних не гарантує покращення продуктивності, оскільки рішення працювали дуже по-різному залежно від схеми.

У роботі [4] було досліджено продуктивність роботи NoSQL баз даних Redis, MongoDB, Memcached, Cassandra, H2. У результаті експериментів був зроблений наступний висновок, що використання пам'яті не є основним критерієм для оцінки продуктивності алгоритмів, оскільки ці бази даних обслуговують дані з пам'яті. Також було досліджено яка з наведених баз даних використовує пам'ять найефективніше. Згідно з результатами, наведеними в цій статті, не існує бази даних, яка забезпечує найкращу продуктивність для всіх операцій з даними. У цій роботі було доведено, що хоча реляційна база даних зберігає свої дані в пам'яті, її загальна продуктивність гірша, ніж у баз даних NoSQL.

У роботі [5] також було досліджено роботу реляційних та нереляційних баз даних та зроблено висновок що зі зростанням об'ємів даних реляційні бази даних програють в продуктивності не реляційним.

Також дослідженням нереляційних баз даних займається багато вітчизняних науковців так наприклад, ще в 2023 році в роботі проф. Мороза Б.І. [6] (в співавторстві) були розглянуті питання автоматизованої класифікації текстових документів, за рахунок використання технології Big Data було доведено зачне підвищення продуктивності роботи системи. У роботі [7] була зроблена загальна характеристика найбільш поширених типів нереляційних баз даних, а також виконана їхня класифікація за типами моделі даних, що використовуються, та нефункціональними характеристиками, базовими з яких є узгодженість даних, доступність та стійкість до розподілу. У статі [8] було досліджено продуктивність роботи баз даних різного типу, для дослідження продуктивності було розроблено спеціальний програмний продукт завдяки якому саме і проводилось дослідження. Однак для того, щоб розібратись чому бази даних потрібно проаналізувати архітектуру та промоделювати процес читання даних.

Мета статті. зробити порівняння архітектури та зробити аналіз швидкодії роботи серверів баз даних на прикладі NoSQL бази даних HBase та СУБД Oracle з різною архітектурою.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження.

Сучасна СУБД Oracle може використовуватися як у режимі Grid так і у режимі Cloude де використовується контейнерна технологія.

Якщо використовується Grid сервери за допомогою високошвидкісної мережі з'єднані між собою та з'єднані з мережевими накопичувачами. [9]

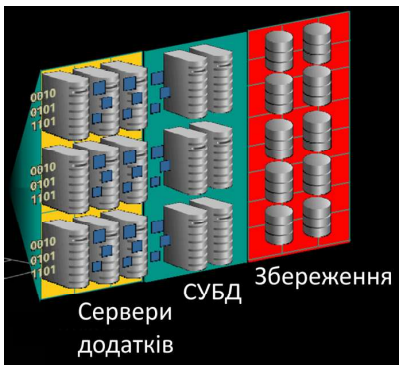

Рис. 1 Архітектура Oracle у режимі Grid

У архітектурі Oracle Grid користувач може з'єднатись з будь-яким сервером для роботи з даними. У цьому режимі інформація про дані зберігається в GRD (Global Resource Directory) таблиці, в якій постійно фіксується, на якому екземплярі, який блок (або його копія) доступний, режим, в якому екземпляр утримує блок, роль блоку та SCN. Коли потрібно прочитати данні з будь якого блоку бази даних запит з початку звертається до цієї таблиці щоб переглянути в якому стані знаходиться цей блок і на якому з серверів. Блок може знаходитись як на жорсткому диску так і в кеші. Якщо якась транзакція працює з даними цього блоку на нього може бути вже накладено якесь блокування. Якщо на блок не накладено блокування транзакція зчитує данні з цього блоку, та накладає відповідно до своїх дій блокування. Його копія переноситься в кеш сервера до якого звернулась транзакція і потім вона саме працює з копією цього блока, і якщо дані змінювались то фоновий процес DBWn та ASM робить запис зміненого блоку на диск. Для прискорення пошуку даних в Grid системах використовуються індекси вони визначають саме номера блоків в яких знаходяться потрібні данні.

Промоделюємо цей процес. Позначимо t_i – час пошуку посилання на блоки у індексі, tз – час зчитування записів у блоці, t_l – час перегляду одного елемента в індексі, t_b – час зчитування одного запису в блоці, n – число знайдених посилань у індексі, N – число записів у блоці. Час зчитування даних і розміщення їх у кеш пам'яті серверу запишеться у вигляді:

$$t_з = D + F\frac{N}{n} \quad (1)$$

де D та F константи які залежать від швидкодії жорстко диску та розміру файлу даних. Аналогічно час доступу до індексу t_i представимо у вигляді:

$$t_i = D_i + F_i n. \quad (2)$$

Щоб знайти відповідний блок потрібно звернутись до GRD таблиці, щоб взнати на якому саме з серверів чи мережевих накопичувачів саме знаходиться цей блок. Час пошуку даних в GRD таблиці позначимо як t_{grd} .

Для оцінки швидкодії припустимо що дані зчитуються у перше та знаходяться на накопичувачі. Тоді час зчитування записів з серверу можна записати у наступному вигляді:

$$T_{grd}(n) = D_i + F_i n + \frac{1}{2}(n+1)t_1 + D + F\frac{N}{n} + \frac{1}{2}\left(\frac{N}{n}+1\right)t_b + nt_{grd}. \quad (3)$$

Якщо блок знаходився вже в кеші і фізичне зчитування з накопичувача буде відсутнє, то тоді формулу можна записати у наступному вигляді:

$$T(n) = D_i + F_i n + \frac{1}{2}\left(\frac{N}{n}+1\right)t_c + nt_{grd} \quad (4)$$

де t_c час зчитування даних з кешу.

Починаючи з 12 версії Oracle використовується мультиконтейнерна архітектура. (Oracle Database Architecture). [10]

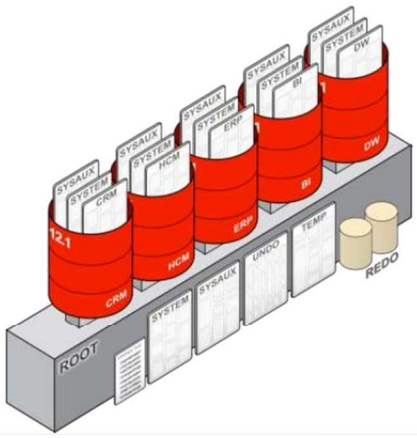

Рис. 2 Архітектура контейнерної бази даних [10]

Особливості мультиконтейнерної архітектури полягає у тому, що багато БД (до 252 баз даних) під'єднані до однієї бази даних, яка є контейнером. Контейнерна база даних керує базами даних що до неї під'єднані. Контейнерна база даних відповідає за розподіл ресурсів, резервне копіювання і відновлення та інше. Обмін даними між серверами проводиться за рахунок лінків. Час

зчитування даних з інших серверів позначимо як t_{link}. У цьому випадку формулу 3 для мультиконтейнерної бази даних можна записати у наступному вигляді:

$$T_{mc}(n) = D_i + F_i n + \frac{1}{2}(n+1)t_1 + D + F\frac{N}{n} + \frac{1}{2}\left(\frac{N}{n}+1\right)t_b + nt_{link}. \quad (5)$$

Час $t_{link} >> t_{grd}$ тому що по-перше GRD таблиця гріду знаходиться в кеші тому час доступу до інформації, про те в якому блоці і на якому з серверів знаходяться данні дуже малий, по-друге коли запит використовує link він під'єднується до віддалених серверів в свою чергу на віддалених серверах створюється сеанси перевіряються права доступу ведеться пошуки у індекс таблицях де саме в яких блоках знаходяться данні. Саме тому

$$T_{grd}(n) < T_{mc}(n). \quad (6)$$

Якщо транзакція працює з даними які знаходяться тільки на одному сервері і $t_{link} = 0$ то тоді навпаки.

$$T_{grd}(n) > T_{mc}(n). \quad (7)$$

HBase - розподілена стовпцово-орієнтована база даних, побудована на основі HDFS. HBase входить до Hadoop, та використовується в ситуаціях, коли необхідно організувати швидкий доступ для читання/запису до дуже великих наборів даних у реальному часі. Система HBase в основному складається з ZooKeeper, HMaster та HRegionServer (рис. 3). HBase зберігає дані через API файлової системи Hadoop. Існує кілька реалізацій інтерфейсу файлової системи – для локальної файлової системи системи, для файлової системи KFS, Amazon S3 та HDFS. Більшість користувачів HBase зберігає дані в HDFS, хоча за умовчанням і без явної вказівки. HBase записує дані у локальну файлову систему. [11]

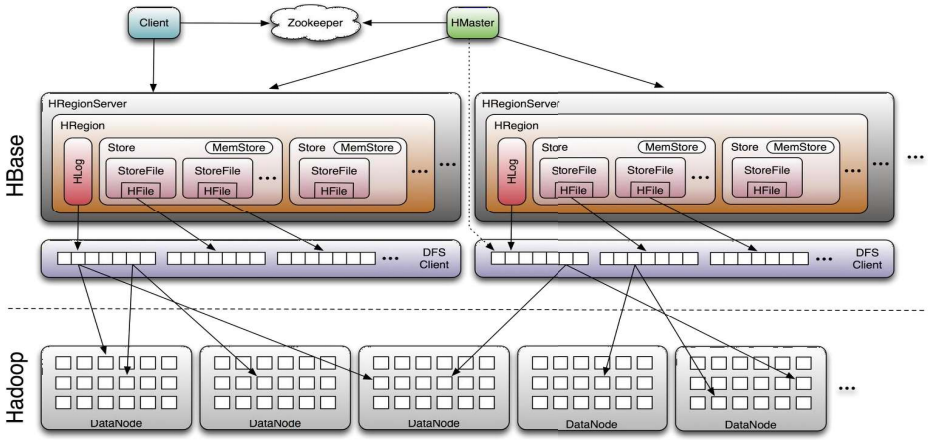

Рис. 3 Архітектура HBase (Sharing On Hbase) [11]

ZooKeeper зберігає метадані, забезпечує точку входу для визначення регіону та контролює HRegionServer. Його механізм голосування більшістю гарантує, що один головний сервер активний і немає єдиної точки відмови.

HMaster керує операціями CRUD над даними, балансуванням навантаження HRegionServer, розподілом регіонів та міграцією регіонів на інші HRegionServer.

HRegionServer зберігає та керує локальним регіоном. Це основний модуль HBase, який відповідає на запити вводу/виводу користувачів та взаємодію даних з HDFS. Усередині хоста HRegionServer є кілька HRegionів, кожен з яких відповідає регіону та складається з кількох сховищ. Кожне сховище відповідає сімейству стовпців. Storefile, Hfile зберігає бінарний файл Hadoop. Storefile — це легка обгортка навколо Hfile. [11]

HBase поділяє таблиці на регіони. Кожен регіон зберігає дані таблиці. При досягненні таблиці великого розміру автоматично створюється новий регіон. В свою чергу регіон складається з кількох сховищ. Кожне сховище відповідає за збереження сімейства стовпців.

Розглянемо яким чином проводиться запис у Hbase. Запис у Hbase проводиться за допомогою менеджера ZooKeeper, система знаходить таблицю -ROOT-, а потім таблицю META, яка зберігає позицію регіону. Потім клієнт використовує цю інформацію для доступу до HRegionServer та відповідного регіону. MemStore – це буфер запису в оперативній пам'яті. Коли він заповнюється,

дані будуть записані в постійне сховище. Дані в MemStore можуть бути втрачені у разі збою сервера. [11]

Читання в Hbase проходить таким чином. Менеджер ZooKeeper за допомогою таблиць -ROOT- та таблиць META визначає регіон в якому зберігаються данні. Після визначення регіону система зчитує у MemStore інформацію про те де зберігаються данні. Якщо дані вже зчитувались вони знаходяться в кеші і система зчитує їх з BlockCache. Якщо дані не знайдено, система шукатиме в постійному сховищі DFS. Промоделюємо цей процес позначимо швидкість пошуку інформації про знаходження даних ZooKeeper - t_{zk}, швидкість зчитування даних з розподіленої файлової системи DFS t_{dfs}. Тоді час зчитування n записів буде дорівнюватись.

$$T_{HB}(n) = t_{zk} + t_{dfs} \quad (8)$$

Данні з таблиць -ROOT- та META після запуску системи зчитуються та зберігаються у кеші тому час пошуку інформації про знаходження даних дуже малий. Системи, до яких входить Hbase, наприклад Hadoop використовують розподілену файлову систему для підвищення швидкості зчитування, файл записується на декілька (K) дисків з яких проводиться зчитування одночасно за рахунок цього в K разів підвищується швидкість. Час зчитування даних і розміщення їх у кеш пам'яті системи запишеться у вигляді:

$$t_{dfs} = D + F\frac{n}{K} \quad (9)$$

де D та F константи які залежать від швидкодії жорстко диску та розміру файлу даних.

$$T_{HB}(n) = t_{zk} + D + F\frac{n}{K} \quad (10)$$

У результаті проведеного моделювання ми бачимо що $T_{mc}(n) > T_{HB}(n)$ та $T_{grd}(n) > T_{HB}(n)$.

У результаті дослідження було доведено, що час зчитування n записів з розподіленої NoSql бази даних (для прикладу було вибрано базу даних Hbase), набагато меншій ніж у реляційних базах даних (для прикладу було вибрано базу даних Oracle з різними типами архітектури). Отже стовпцовоорієнтовані хоча Hbase не підтримує транзакції, не підтримує обмежень цілісності, як реляційні бази даних, однак мають дуже високу швидкість доступу до даних, що саме і забезпечує їх роботу з даними великого розміру.

Висновки та перспективи подальшого дослідження. У результаті проведеного дослідження було доведено на прикладі реляційних баз даних з різними конфігураціями на прикладі СУБД Oracle та стовпчико-орієнтованою базою даних Hbase, що не зважаючи на недоліки стовпчико-орієнтованої бази даних швидкість доступу до даних в неї швидчий за рахунок її архітектури. Однак за рахунок того що стовпчикоорієнтовані бази даних мають значні недоліки а саме вони не підтримують обробку транзакцій та обмежень цілісності сфера їх застосування дуже обмежена. В подальшому можна провести дослідження об'єктоорієтованих баз даних та порівняти швидкість їх роботи з реляційними СУБД та базами даних NoSql.

Список бібліографічного опису

1. Chen Yang Theoretical Analysis of Distributed Systems and Their Scalability Advances in Computer, Signals and Systems (2025) Clausius Scientific Press, Canada Vol. 9 Num. 1 DOI: 10.23977/acss.2025.090104
2. Henrique D. Lima [a], Luiz A. de P. Lima Jr. [a], Alcides Calsavara [a], Henri F. Eberspächer [a], Ricardo C. Nabhen [a], Elias P. Duarte Jr.Beyond scalability: Swarm intelligence affected by magnetic fields in distributed tuple spaces Journal of Parallel and Distributed Computing Volume 123, January 2019, P. 90-99. https://doi.org/10.1016/j.jpdc.2018.09.004
3. Ferreira, F. and Fidalgo, R. A DbaaS exempts the user from upfront investments and allows cost optimization, since many DBaaS solutions provide dynamic cloud resource allocation (Idrissi, 2016). DOI: 10.5220/0012546200003690
4. Abdullah Talha Kabakusa, ResulKara Aperformanceevaluationof in-memorydatabases. // JournalofKingSaudUniversity–Computer and Information Sciences (2017) 29, p. 520–525.
5. Roman Čerešňák*, Michal Kvet Comparison of query performance in relational a non-relation databases // 13th International Scientific Conference on Sustainable, Modern and Safe Transport (TRANSCOM 2019), High Tatras, Novy Smokovec – Grand Hotel Bellevue, Slovak Republic, May 29-31, 2019.
6. Тарасюк О.М., Таранова К.П., Горбенко А.В. / Аналіз особливостей та класифікація систем управління нереляційними базами даних // Системи управління, навігації та зв'язку, 2017, випуск 4(44). С. 116-118.

7. Мороз, Б., Кабак, Л., Варех, Н., Мороз, Д. (2023). Система класифікації текстових документів із використанням технологій Big Data. Information Technology: Computer Science, Software Engineering and Cyber Security, 2, с. 34–40. doi: https://doi.org/10.32782/IT/2023-2-4
8. Нікітіна Т. С., Морозова О. І. / Порівняльний аналіз продуктивності баз даних SQL та NoSQL // Системи управління, навігації та зв'язку, 2019, випуск 1(53) с. 125 -128.
doi: 10.26906/SUNZ.2019.1.125
9. Oracle Grid Computing Електронний ресурс. URL: https://docs.oracle.com/cd/E11882_01/rac.112/e41960/admcon.htm#i1058057
10. Oracle Database Architecture Електронний ресурс. URL: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/cncpt/introduction-to-oracle-database.html#GUID-8F2EEEC8-0372-4419-88FF-7D77A9C0FCAD
11. Sharing On Hbase Електронний ресурс. URL: https://vhida.github.io/Sharing-on-HBase/

References

1. Chen Yang Theoretical Analysis of Distributed Systems and Their Scalability Advances in Computer, Signals and Systems (2025) Clausius Scientific Press, Canada Vol. 9 Num. 1 DOI: 10.23977/acss.2025.090104
2. Henrique D. Lima [a], Luiz A. de P. Lima Jr. [a], Alcides Calsavara [a], Henri F. Eberspächer [a], Ricardo C. Nabhen [a], Elias P. Duarte Jr.Beyond scalability: Swarm intelligence affected by magnetic fields in distributed tuple spaces Journal of Parallel and Distributed Computing Volume 123, January 2019, P. 90-99. https://doi.org/10.1016/j.jpdc.2018.09.004
3. Ferreira, F. and Fidalgo, R. A DbaaS exempts the user from upfront investments and allows cost optimization, since many DBaaS solutions provide dynamic cloud resource allocation (Idrissi, 2016). DOI: 10.5220/0012546200003690
4. Abdullah Talha Kabakusa, ResulKara Aperformanceevaluationof in-memorydatabases. // JournalofKingSaudUniversity–Computer and Information Sciences (2017) 29, p. 520–525.
5. Roman Čerešňák*, Michal Kvet Comparison of query performance in relational a non-relation databases // 13th International Scientific Conference on Sustainable, Modern and Safe Transport (TRANSCOM 2019), High Tatras, Novy Smokovec – Grand Hotel Bellevue, Slovak Republic, May 29-31, 2019.
6. Tarasiuk O.M., Taranova K.P., Horbenko A.V. / Analiz osoblyvostei ta klasyfikatsiia system upravlinnia nereliatsiinymy bazamy danykh // Systemy upravlinnia, navihatsii ta zviazku, 2017, vypusk 4(44). C. 116-118.
7. Moroz, B., Kabak, L., Varekh, N., Moroz, D. (2023). Systema klasyfikatsii tekstovykh dokumentiv iz vykorystanniam tekhnolohii Big Data. Information Technology: Computer Science, Software Engineering and Cyber Security, 2, с. 34–40. doi: https://doi.org/10.32782/IT/2023-2-4
8. Nikitina T. S., Morozova O. I. / Porivnialnyi analiz produktyvnosti baz danykh SQL ta NoSQL // Systemy upravlinnia, navihatsii ta zviazku, 2019, vypusk 1(53) s. 125 -128.
doi: 10.26906/SUNZ.2019.1.125
9. Oracle Grid Computing URL: https://docs.oracle.com/cd/E11882_01/rac.112/e41960/admcon.htm#i1058057
10. Oracle Database Architecture URL: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/cncpt/introduction-to-oracle-database.html#GUID-8F2EEEC8-0372-4419-88FF-7D77A9C0FCAD
11. Sharing On Hbase Електронний ресурс. URL: https://vhida.github.io/Sharing-on-HBase/