УДК 519.688 https://doi.org/10.31713/vt4202012

Бачишина Л. Д., к.е.н., доцент, Харів Н. О., ст. викладач, Оссас М. О., студент (Національний університет водного господарства та природокористування, м. Рівне)

ЗАСТОСУВАННЯ СТРАТЕГІЇ NOSQL ДЛЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВІДДІЛУ КАДРІВ ПІДПРИЄМСТВА

Обсяги інформації, які зараз передаються через Інтернет, обумовили появу технології Big Data. Довгий час в інформаційних системах для представлення інформації використовувалась реляційна модель даних. Але ця модель не завжди може задовольняти сучасні потреби користувачів. Все популярнішими стають нереляційні бази даних. Метою цієї роботи є порівняння реляційної і нереляційної баз даних та визначення: яка з них підходить для вирішення поставленої задачі. Досліджено особливості роботи і використання нереляційних баз даних у порівнянні з реляційними. Вивчено переваги та недоліки використання нереляційних СКБД. Встановлено, що основною перевагою NoSQL є її гнучкість, краща масштабованість, швидкість обробки запитів, можливість змінювати структуру документів з мінімальними переробками програмного коду. Визначено стратегію, яка найкраще підходить для розробки інформаційної системи відділу кадрів на виробництві. Проект реалізовано з використанням нереляційної документо-орієнтованої системи керування базами даних MongoDB. Встановлено, що перевагою MongoDB є не лише можливість створення різноманітних зв'язків між документами, але й використання вбудованих документів, що спрощує організацію запитів до сховища даних.

***Ключові слова*: бази даних; SQL; NoSQL; MongoDB; CRUD – операції.**

Вступ. У сучасному світі із швидким розвитком науки, техніки та сучасних технологій нагально постає проблема зберігання, обробки та використання великих обсягів інформації. Незважаючи на широке застосування і постійний розвиток можливостей, реляційні бази даних не завжди можуть задовольняти сучасні потреби користувачів. Величезні обсяги інформації, які зараз передаються через Інтернет, обумовили появу технології Big Data. У зв'язку з цим виникла необхідність у стратегії, яка має більшу швидкість доступу і масштабованість. Найкращим рішенням є застосування нереляційних, так званих

NoSQL баз даних, які дістали значного розвитку та поширення впродовж кількох останніх років і широко застосовуються при створенні розподілених інформаційних систем, які повинні зберігати та швидко обробляти великі обсяги інформації.

Метою даної роботи є створення NoSQL бази даних та інформаційної системи відділу кадрів підприємства, а також дослідження застосування нереляційних баз даних і порівняння їх використанням SQL у реляційних базах даних.

Можливості NoSQL та засоби розробки нереляційних баз даних. Концепція нереляційних баз даних застосовувалась для обробки та зберігання даних до появи SQL. Довгий час домінуючу позицію на ринку займали реляційні бази даних. І зараз значна частина розробників використовує їх. Згідно рейтингу DB-Engines Ranking [1], реляційні бази даних залишаються досить популярними. Структура реляційних баз даних концептуально проста та зрозуміла. Вона дозволяє створювати невеликі за обсягом бази даних, легко і швидко їх обробляти. Більшість сучасних комерційних СКБД використовують реляційну модель бази даних. Проте, великі потоки інформації призвели до появи хмарних технології та Dig Data. У зв'язку з цим застосування нереляційних баз даних принесло з собою безліч цікавих і комерційно успішних ідей для потреб різних компаній та проєктів. Не можна стверджувати, що реляційні бази даних приречені. Швидше за все, у процесі еволюції зберігання і обробки даних з'являться комбіновані рішення, де NoSQL системи будуть використовуватись, щоб усунути слабкі місця SQL.

В роботі [2] автори відзначають, що перевагами реляційних баз даних є відповідність ACID та підтримка цілісності даних. Але, цей тип баз даних потребує потужних серверів для масштабованості, структура бази даних часто буває складною і важкою для обробки. Реляційні СКБД потребують дорогих систем зберігання. Тому, коли мова іде про великі обсяги погано структурованих даних, варто віддати перевагу нереляційним базам даних.

Виділяють чотири основних види NoSQL баз даних: «ключ-значення», документо-орієнтовані бази даних, графова модель бази даних, bigtable-подібні бази даних. Бази даних «ключ-значення» цікава в першу чергу компаніям, що надають хмарні послуги. Найбільш відомі СУБД для даного типу баз даних – це Amazon DynamoDB, Berkeley DB, MemcacheDB, Redis і Riak. Документо-орієнтована БД являє собою набір ієрархічних, деревоподібних структур даних (документів). Цей тип баз даних підійде до задач, де потрібно впорядко-

ване зберігання інформації, але немає безлічі зв'язків між даними і немає потреби постійно збирати статистику по них. Серед СКБД даного типу найбільш популярними є CouchDB, Couchbase, MarkLogic, MongoDB.

Зараз лідером на ринку серед NoSQL рішень є MongoDB. Це документо-орієнтована система керування базами даних, яка широко використовується JavaScript розробниками. Це пов'язано з тим, що дані всередині документів (аналог таблиць у реляційних СКБД) зберігаються у JSON-подібному форматі, який є звичним як для JavaScript, так і для сфери веброзробки в цілому. Ще однією особливістю, яка полегшує життя розробникам є можливість створювати не лише різноманітні зв'язки між документами, але й вбудовані документи, що спрощує запити до сховища даних. Графові бази даних найкраще підходять для реалізації проєктів, які передбачають природну графову структуру даних. Насамперед це соціальні мережі. У подібних завданнях цей тип даних сильно випереджає реляційні по продуктивності, простоті внесення змін і наочності подання інформації. Найбільш відомі графові СУБД – це Neo4j, ArangoDB, FlockDB, HyperGraphDB, OrientDB. Bigtable-подібні бази даних містять дані, впорядковані у вигляді розрідженої матриці, рядки і стовпці якої використовуються в якості ключів. Такі бази даних застосовуються для вебіндексування та розв'язування інших завдань, які передбачають роботу з величезними обсягами даних. Прикладами СУБД даного типу є HBase, Cassandra, Hypertable, SimpleDB.

Зараз лідером на ринку серед NoSQL рішень є MongoDB. Це документо-орієнтована система керування базами даних, яка широко використовується JavaScript розробниками. Насамперед, це пов'язано з тим, що дані всередині документів (аналог таблиць у реляційних СКБД) зберігаються у JSON-подібному форматі, який є звичним як для JavaScript, так і для сфери веброзробки в цілому. Ще однією особливістю, яка полегшує життя розробникам є можливість створювати не лише різноманітні зв'язки між документами, але й вбудовані документи, що спрощує запити до сховища даних. Оскільки робота у відділі кадрів передбачає обробку великої кількості документів, для проєкту було обрано саме MongoDB.

Програмна реалізація інформаційної системи відділу кадрів. Серверна частина додатку написана на мові програмування JavaScript, під сервер Node.js. Каркасом додатку служить фреймворк Express.js. В ролі сховища даних виступає MongoDB, для роботи з якою використовується ODM (Object Data Model) Mongoose. Це бібліо-

тека, яка зв'язує базу даних із сервером. UI (user interface) створено у вигляді вебдодатка, написаного з використанням React.js і SemanticUI. Даний стек технологій дозволяє ефективно і швидко розробляти якісні вебдодатки, а тому має високий попит на ринку праці. Варто зазначити, що цей стек потребує знання лише однієї мови програмування – JavaScript.

Комунікація між сервером і клієнтським додатком здійснюється за допомогою прикладного програмного інтерфейсу, відомого як api. Сервер має точки доступу (endpoint), при зверненні до яких виконується відповідна бізнес логіка частиною якої є взаємодія з базою даних. На початку роботи потрібно авторизуватися через форму входу (рис. 1).

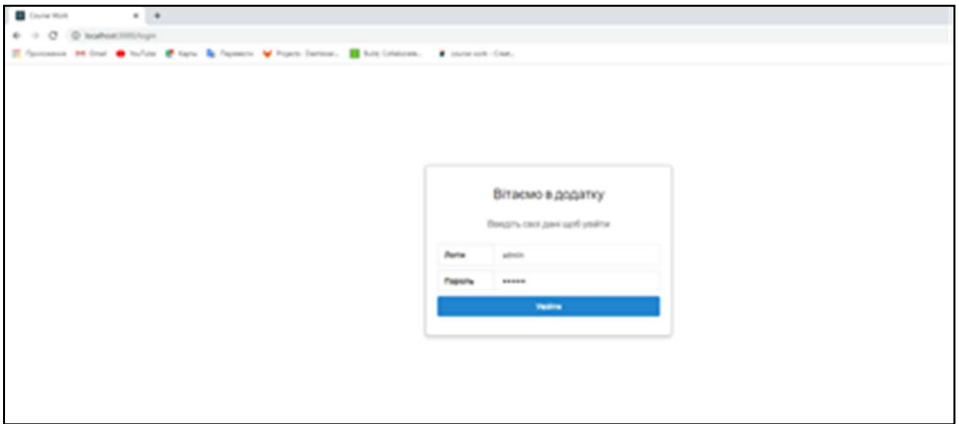

Рис. 1. Форма входу

Увійти до додатка відразу може лише адміністратор, який надає доступ іншим працівникам, вказавши при цьому, чи можуть вони виконувати CRUD операції в додатку. Таким чином створюються нові записи в документі Credentials, у якому зберігаються дані для входу в додаток. Після успішної аутентифікації надається доступ до додатку, у якому користувач може продовжити роботу із відділами та працівниками.

В базі даних використовуються чотири колекції: відділення, посади, працівники та дані для авторизації (credentials), схеми яких мають вигляд відображений на рис. 2–5.

```
const credentialsSchema = new mongoose.Schema({
    login: String,
    password: String,
    allowEdit: Boolean,
    user: { type: Schema.Types.ObjectId, ref: 'Workers' },
});
```

Рис. 2. Схема даних для авторизації

```
var departmentSchema = new mongoose.Schema({
  name: { type: String, required: true },
  address: String,
  workers: [{ type: Schema.Types.ObjectId, ref: 'Workers'}],
  head: { type: Schema.Types.ObjectId, ref: 'Workers'}
});
```

Рис. 3. Схема відділення

```
const workerSchema = new mongoose.Schema({
  name: String,
  birth: Date,
  hired: Date,
  position: { type: Schema.Types.ObjectId, ref: 'Positions' },
  subordinates: [{ type: String, ref: 'Workers' }],
  head: { type: String, ref: 'Workers' },
  creds: { type: Schema.Types.ObjectId, ref: 'Credentials' },
});
```

Рис. 4. Схема працівника

```
const positionsSchema = new mongoose.Schema({
  name: String,
  defaultSalary: Number
});
```

Рис. 5. Схема посади

EER-схема відповідної реляційної бази даних мала б вигляд, що поданий на рис. 6.

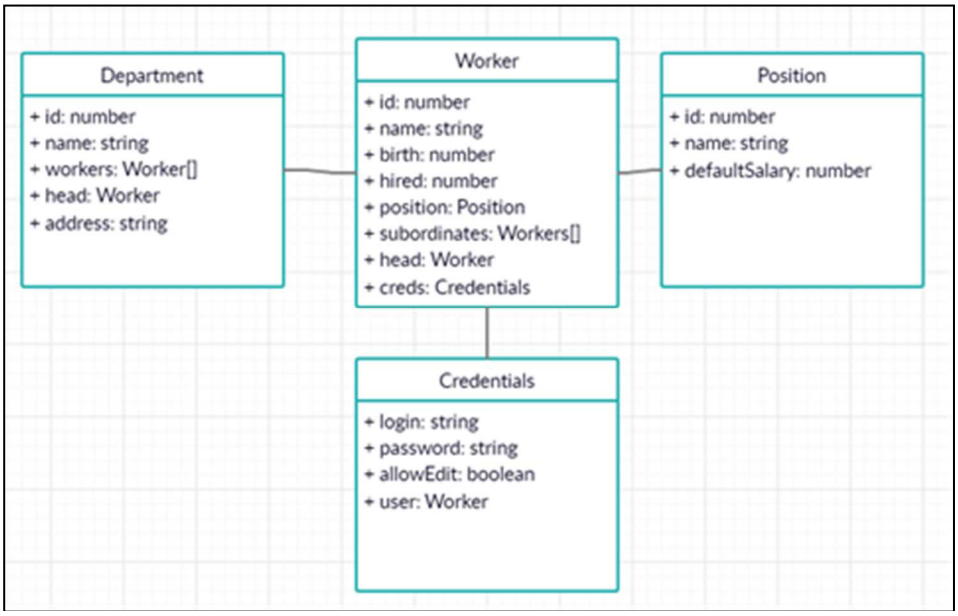

Рис. 6. Схема даних для реляційної бази даних

Кожен документ колекції має власний ідентифікатор _id, який створюється автоматично (_id: mongoose.Schema.Types.ObjectId). Головною колекцією документів є Department, де зберігається інформація про підрозділи підприємства, а також список працівників, які до нього належать. У документах колекції працівника (Worker) зберігається загальна інформацію про працівників та їх підлеглих (subordinates). Якщо працівник є чиїмось підлеглим, то вказується посилання на керівника (head). Колекція Positions зберігає загальну інформацію про доступні посади. Користувачеві надається можливість переглядати, фільтрувати та коригувати інформацію про працівників обраного підрозділу. Адміністратор бази даних може створювати нових працівників і надавати їм доступ до додатка. Для зручності роботи з великою кількістю працівників, вибірку можна фільтрувати за допомогою кнопок, вибравши критерії для фільтрації.

SQL-операторам створення, читання, знищення та коригування об'єктів в нереляційних базах даних відповідають CRUD-операції [5]. Наступні приклади (рис. 7–10) ілюструють виконання CRUD-операцій. На рис. 7 відображено код функції створення нового відділу. При створенні виконується перевірка, чи існує відділ з введеною назвою, і, якщо він відсутній, то створюється новий за допомогою конструктора new DepartmentModal ({ name, address, workers: [], head: null }). Збереження інформації у бази даних відбувається за допомогою ви-

клику метода save, який в якості аргументу отримує функцію зворотного виклику (callback).

```
app.post(apiEndpoints.departments.addOne, async (req, res) => {
  const { body: { name, address } } = req;
  const allowCreate = await DepartmentsModal.find({ name })
    .then(x => !x.length)
    .catch(y => {
      console.log('error: ', y);
      return null;
    });

  if (!allowCreate) {
    res.status(HttpStatus.INTERNAL_SERVER_ERROR).send();
    return;
  }

  new DepartmentsModal({
    name,
    address,
    workers: [],
    head: null
  }).save((err, newDepartment) => {
    res.status(
      err
      ? HttpStatus.INTERNAL_SERVER_ERROR
      : HttpStatus.ACCEPTED
    ).send(newDepartment);
  });
});
```

Рис. 7. Операція Create

Для отримання даних про усіх користувачів, які належать до одного відділу, написано функцію getAllDepartmentWorkers. Параметрами є id відділу (depId) і список параметрів для фільтрації вибірки (query). Пошук відділення виконується наступним чином: const department = await DepartmentsModal.findOne({ _id: depId }).catch(() => null). Після цього формується список id працівників даного відділу. Пошук здійснюється викликом методу find з переданими параметрами пошуку і фільтрації: WorkerModel.find(condition).catch(() => []).

```
getAllDepartmentWorkers = async (depId, query) => {
  const department = await DepartmentsModal.findOne({ _id: depId }).catch(() => null);
  const workersIds = department
  ? department.workers
  : [];
  const queryFilters = serializeFilterQuery(query);
  const condition = { '_id': { $in: workersIds }, ...queryFilters };
  return await WorkerModel.find(condition).catch(() => []);
}
```

Рис. 8. Операція Read

В MongoDB оновлення даних представлене доволі гнучкою функцією, яка може повністю змінити запис або створити новий у випадку коли його ще немає, а також оновити або видалити вибране поле/поля чи елементи масиву. В програмі використовується оператор $set, який оновлює не весь документ, а лише дані за значенням одного з його ключів. Однак, якщо йому на вхід передати об'єкт, то він замінить існуючі дані і добавить нові.

```
app.put(apiEndpoints.departments.updateOne, (req, res) => {
  const { params: { id }, body: { name, address } } = req;
  DepartmentsModal.findOneAndUpdate(
    { _id: id},
    { $set: { name, address } },
    { new: true },
    ( err, updatedOne ) => {
      res.status(
        (!!err || !updatedOne)
        ? HttpStatus.INTERNAL_SERVER_ERROR
        : HttpStatus.ACCEPTED
      ).send(updatedOne);
    }
  );
});
```

Рис. 9. Операція Update

В MongoDB є декілька функцій для видалення даних. В даному випадку використовується функція deleteOne з параметром id відділу, за яким здійснюється пошук і видалення відповідної інформації з бази даних.

```
app.delete(apiEndpoints.departments.removeOne, async (res, req) => {
  const { params: { id: _id }} = res;
  const { deletedCount } = await DepartmentsModal.deleteOne({ _id });
  req.status(
    deletedCount
    ? HttpStatus.ACCEPTED
    : HttpStatus.CONFLICT
  ).send();
});
```

Рис. 10. Операція Delete

Для обробки HTTP-запиту існує чотири методи: get, post, put, delete (рис. 11). Для кожного з них відведено своя роль – отримати, створити, оновити, видалити ресурс в перерахованому порядку. В параметри функції вони приймають шлях та будь-яку кількість функцій обробників (handler чи callback). Функцію, що обробляє інформацію і вирішує, чи варто її допустити до наступного обробника, називають функцією посередником (middleware). Кожен такий обробник приймає в параметри запит (req) та відповідь (res), а також метод next(), який власне виконує запуск наступного обробника.

```
app.get(apiEndpoints.departments.getAll, (req, res) => {···
});

app.post(apiEndpoints.departments.addOne, async (req, res) => {···
});

app.put(apiEndpoints.departments.updateOne, (req, res) => {···
});

app.delete(apiEndpoints.departments.removeOne, async (res, req) => {···
});
```

Рис. 11. Функції обробки HTTP-запиту

Висновки. Здійснено порівняння використання реляційних та нереляційних баз даних для розробки вебдодатку відділу кадрів підприємства. Встановлено, що засобами MongoDB можна виконати операції, що мають відповідні аналоги в SQL-запитах. Проте, основною перевагою NoSQL, на наш погляд, є її гнучкість, можливість змінювати структуру документів з мінімальними зусиллями, що особливо важливо для невеликих проєктів. Водночас РСУБД є рішенням, перевіреним роками, NoSQL гарантує все те саме, але з додатковими перевагами. Однією з них є можливість створювати вбудовані доку-

менти, що спрощує формування запитів до сховища даних та прискорює обробку інформації в базах даних, що орієнтовані на роботу з великою кількістю документів.

1. DB-Engines Ranking. URL: https://db-engines.com/en/ranking (accessed date: 20.01.2021). **2.** Швець М., Заруба Д. С., Хохлов Ю. В. Порівняння SQL та NOSQL баз даних. *Вчені записки ТНУ імені В.І. Вернадського*. *Сер. Технічні науки*. 2018. № 6. Том 29 (68). Ч. 2. С. 21–25. URL: http://www.tech.vernadskyjournals.in.ua/journals/2018/6_2018/part_2/ 7.pdf (дата звернення: 20.01.2021). **3.** Эрик Рэдмонд, Джим Р. Уилсон. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL. М. : ДМК Пресс, 2013. 384 с. **4.** Кайл Бэнкер. MongoDB в действии. М. : ДМК Пресс, 2012. 394 с. **5.** Фаулер, Мартин, Садаладж, Прамодкумар Дж. NoSQL: новая методология разработки нереляционных баз данных. М. : ООО «И.Д. Вильямс», 2013. 192 с. **6.** Холмс С. Стэк MEAN. Mongo, Express, Angular, Node. СПб. : Питер, 2017. 496 с. **7.** Шеннон Брэдшоу, Йон Брэзил, Кристина Ходоров. MongoDB: полное руководство. М. : ДМК Пресс, 2020. 540 с.

REFERENCES:

1. DB-Engines Ranking. URL: https://db-engines.com/en/ranking (accessed date: 20.01.2021). **2.** Shvets M., Zaruba D. S., Khokhlov Yu. V. Porivniannia SQL ta NOSQL baz danykh. *Vcheni zapysky TNU imeni V.I. Vernadskoho*. *Ser. Tekhnichni nauky*. 2018. № 6. Tom 29 (68). Ch. 2. S. 21–25. URL: http://www.tech.vernadskyjournals.in.ua/journals/2018/6_2018/part_2/ 7.pdf (data zvernennia: 20.01.2021). **3.** Erik Redmond, Djim R. Uilson. Sem baz dannyih za sem nedel. Vvedenie v sovremennyie bazyi dannyih i ideologiyu NoSQL. M. : DMK Press, 2013. 384 s. **4.** Kayl Benker. MongoDB v deystvii. M. : DMK Press, 2012. 394 s. **5.** Fauler, Martin, Sadaladj, Pramodkumar Dj. NoSQL: novaya metodologiya razrabotki nerelyatsionnyih baz dannyih. M. : OOO «I.D. Vilyams», 2013. 192 s. **6.** Holms S. Stek MEAN. Mongo, Express, Angular, Node. SPb. : Piter, 2017. 496 s. **7.** Shennon Bredshou, Yon Brezil, Kristina Hodorov. MongoDB: polnoe rukovodstvo. M. : DMK Press, 2020. 540 s.

Bachyshyna L. D., Candidate of Economics (Ph.D.), Associate Professor, Khariv N. O., Senior Lecturer, Ossas M. O., Senior Student (National University of Water and Environmental Engineering, Rivne)

NOSQL STRATEGY APPLICATION TO DESIGN INFORMATION SYSTEM OF THE ENTERPRISE HUMAN RESOURCES DEPARTMENT

The amount of information, which is transmitted by the Internet, brought to the emergence of Big Data technology. The relational data model was used into information systems to represent information for a long time. Unfortunately, this model not always correspond to the modern user's needs. Non-relational databases are becoming more popular than the relation databases. The purpose of this research is to compare relational and non-relational databases and determine which one is more suitable for the task set. The peculiarities of work and use a non-relational database was investigated in comparison with relational ones. The advantages and disadvantages of using non-relational DBMS was studied. It had been found that the main advantage of NoSQL is flexibility, better scalability, speed of query processing, the ability to change the structure of documents with minimal code alteration. The best strategy that is suited for the development of an information system for the personnel department in production has been determined. The project was implemented using MongoDB, a non-relational document-oriented database management system. It was detected that MongoDB has the advantage of being able to create not only a variety of links between documents, but also embedded documents, simplifies the organization of queries to the data store.
***Keywords*: databases; SQL, NoSQL; MongoDB; CRUD-operations.**

Бачишина Л. Д., к.э.н., доцент, Харив Н. А., ст. преподаватель, Оссас М., студент (Национальный университет водного хозяйства и природопользования, г. Ровно)

ПРИМЕНЕНИЕ СТРАТЕГИИ NOSQL ДЛЯ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ ОТДЕЛА КАДРОВ ПРЕДПРИЯТИЯ

Объемы информации, которые сейчас передаются через Интернет, обусловили появление технологии Big Data. Долгое время в ин-

формационных системах для представления информации использовалась реляционная модель данных. Но данная модель не всегда может соответствовать современным потребностям пользователей. Все популярнее становятся нереляционные базы данных. Целью этой работы является сравнение реляционной и нереляционной баз данных и определения какая из них больше подходит для решения поставленной задачи. Исследованы особенности работы и использования нереляционных баз данных в сравнении с реляционными. Изучены преимущества и недостатки использования нереляционных СУБД. Установлено, что основным преимуществом NoSQL является ее гибкость, лучшая масштабируемость, скорость обработки запросов, возможность изменять структуру документов с минимальными переделками кода. Определена стратегия, которая лучше всего подходит для разработки информационной системы отдела кадров на производстве. Проект реализован с использованием нереляционной документо-ориентированной системы управления базами данных MongoDB. Установлено, что преимуществом MongoDB является не только возможность создания разнообразных связей между документами, но и использование встроенных документов, что упрощает организацию запросов к хранилищу данных.

***Ключевые слова*: базы данных; SQL; NoSQL; MongoDB; CRUD-операции.**
