NoSQL i upravljanje nerelacijskim podacima

Ishodi učenja

Nakon završetka ove nastavne cjeline studenti će moći:

- objasniti temeljne koncepte, razloge primjene i arhitekturne razlike između relacijskih i nerelacijskih (NoSQL) sustava baza podataka te njihove prednosti i ograničenja u suvremenim podatkovnim okruženjima;
- razvrstati i opisati glavne vrste NoSQL baza podataka (ključ–vrijednost, dokumentne, sa skupinama stupaca i grafovske baze) te prepoznati odgovarajuće primjene i reprezentativne tehnologije za svaku vrstu;
- razumjeti i vrednovati NoSQL rješenja u programskim arhitekturama, osobito u kontekstu velikih podataka, računarstva u oblaku i analitike u stvarnom vremenu, radi odabira učinkovitih i fleksibilnih strategija upravljanja podacima.

1. Uvod: informacijski sustavi okoliša s intenzivnom obradom podataka

1.1. Nagli rast količine podataka o okolišu

Tijekom posljednja dva desetljeća praćenje stanja okoliša doživjelo je temeljitu tehnološku preobrazbu. Suvremeni ekološki sustavi prate se gustim mrežama IoT senzora, satelitskim konstelacijama, autonomnim bespilotnim letjelicama, bazama podataka o bioraznolikosti, klimatskim modelima i platformama građanske znanosti. Takvi sustavi neprekidno stvaraju goleme količine raznorodnih podataka o atmosferskim uvjetima, hidrološkim ciklusima, promjenama korištenja zemljišta, obrascima bioraznolikosti i pokazateljima stanja ekosustava.

Za razliku od tradicionalnih podataka o okolišu, koji su se prikupljali povremeno i naknadno obrađivali, suvremena infrastruktura praćenja stvara podatkovne tokove velike brzine, količine i raznolikosti. Već jedna regionalna mreža za praćenje kvalitete zraka može mjesečno generirati milijune vremenski označenih mjerenja. Satelitske platforme svakodnevno isporučuju terabajte rasterskih podataka. Portali bioraznolikosti kao što je GBIF pohranjuju stotine milijuna zapisa o pojavljivanju vrsta, zajedno s prostornim i vremenskim metapodacima.

Ta je preobrazba dovela do razvoja informacijske infrastrukture za intenzivnu obradu podataka o okolišu. Takvi sustavi moraju:

- objedinjavati raznorodne strukturirane i nestrukturirane podatke;
- raditi u geografski distribuiranim okruženjima;
- podržavati analitiku u gotovo stvarnom vremenu;
- omogućiti elastično povećavanje kapaciteta;
- ostati otporni na nepredvidive događaje u okolišu.

Tradicionalne podatkovne arhitekture nisu bile oblikovane za toliku dinamičnost.

1.2. Od praćenja stanja okoliša do distribuiranih podatkovnih platformi

Suvremene platforme za praćenje okoliša obrađuju skupove koji povezuju više vrsta podataka, kao što prikazuje tablica 1.

Tablica 1. Vrste podataka o okolišu

Vrsta podataka	Strukturna obilježja	Primjeri
Vremenski nizovi	Uzastopni, visoke učestalosti	Temperatura, relativna vlažnost, PM2,5, vlažnost tla
Prostorni podaci	Geoprostorni	Koordinate, zaštićena područja, poligoni pokrova zemljišta
Temeljena na događajima	Polustrukturirani	Detekcija divljih životinja, upozorenja na poplave
Metapodaci	Dokumentni	Specifikacije senzora, zapisi o umjeravanju
Podaci o odnosima	Grafovska struktura	Veze vrsta i staništa, mreže slivova
Tekstni podaci	Nestrukturirani	Terenske bilješke, izvješća, anotacije

Istodobna prisutnost tih oblika podataka donosi arhitekturne izazove:

- strogo definirane sheme postaju nepraktične;
- potrebno je često mijenjati podatkovnu shemu;

- relacijski upiti s brojnim operacijama JOIN pri velikom opterećenju gube na učinkovitosti;
- vertikalno povećavanje kapaciteta postaje ekonomski neučinkovito;
- distribuirana replikacija u tradicionalnim RDBMS okruženjima složena je.

Kako praćenje okoliša prelazi na neprekidne, stvarnovremenske i geografski distribuirane sustave, sve je izraženija potreba za fleksibilnim modelima baza podataka koji omogućuju horizontalno skaliranje.

1.3. Ograničenja relacijskih arhitektura

Relacijski sustavi za upravljanje bazama podataka (RDBMS) posebno su uspješni u očuvanju dosljednosti transakcija i radu s jasno definiranim shemama. Međutim, suočavaju se sa sljedećim ograničenjima.

1. Krutost sheme. Sustavi senzorskog praćenja okoliša neprestano se razvijaju uvođenjem novih mjernih parametara, primjerice UV indeksa, saliniteta vode ili koncentracije mikroplastike. Zbog toga su potrebne česte promjene sheme baze podataka, a ponavljane naredbe ALTER TABLE sve više opterećuju stabilnost i učinkovitost sustava.
2. Upiti s brojnim spajanjima. Višedimenzionalne analize okoliša često zahtijevaju povezivanje senzorskih mjerenja, metapodataka o lokacijama, prostornih slojeva i zapisa o vrstama. Operacije JOIN nad milijunima redaka zato mogu znatno usporiti izvođenje upita.
3. Ograničeno horizontalno skaliranje. Povećavanje kapaciteta relacijskih sustava tradicionalno se oslanja na vertikalno skaliranje, odnosno snažniju sklopovsku opremu, što je skupo i ima konačne granice.
4. Neučinkovitost pri radu s prostornim i grafovskim podacima. Premda postoje relacijska proširenja poput PostGIS-a, ona povećavaju složenost i ne skaliraju se uvijek učinkovito u distribuiranim arhitekturama.
5. Izazovi prihvata tokova podataka. Učestali upisi iz podatkovnih tokova mogu preopteretiti mehanizme zaključavanja transakcija.

Ta ograničenja ne znače da su relacijske baze zastarjele. Pokazuju da sustavi za rad s velikim količinama podataka o okolišu trebaju komplementarne ili alternativne arhitekturne pristupe.

Upravo su zato važni NoSQL sustavi i nerelacijsko upravljanje podacima.

NoSQL sustavi predstavljaju prijelaz s projektiranja usmjerenog na unaprijed zadanu shemu prema modeliranju prilagođenom primjeni. Umjesto pitanja „Kako normalizirati podatke?” polazi

se od pitanja „Kako će se podacima pristupati, kako će ih se pretraživati i kako će se sustav skalirati?” Taj pristup počiva na nekoliko načela:

- fleksibilne ili dinamičke sheme;
- denormalizacija kao strategija povećanja učinkovitosti;
- horizontalno skaliranje kao polazna pretpostavka;
- kompromisi između zahtjeva za dosljednošću u distribuiranim sustavima;
- projektno predviđeno particioniranje i repliciranje podataka.

U sustavima za praćenje okoliša, u kojima upiti često obuhvaćaju novija vremenska razdoblja, prostorna područja ili kategorije ekosustava, oblikovanje prema obrascima pristupa podacima postaje važnije od stroge normalizacije.

2. Osnove NoSQL-a i distribuiranih podatkovnih arhitektura

2.1. Teorijska podloga

Pojava NoSQL tehnologija označava znatnu promjenu u načinu promišljanja i razvoja distribuiranih podatkovnih sustava. Relacijske baze oslanjaju se na unaprijed definirane sheme i stroga načela normalizacije, dok suvremeni informacijski sustavi okoliša sve češće rade u dinamičnim, heterogenim i geografski distribuiranim okruženjima u kojima takva krutost predstavlja ograničenje. Baza podataka pritom više nije samo strukturirano spremište nego prilagodljiva sastavnica šire distribuirane arhitekture.

Unatoč međusobnim razlikama, NoSQL baze imaju nekoliko zajedničkih obilježja:

- *Fleksibilnost sheme: NoSQL sustavi često ne zahtijevaju unaprijed definiranu shemu ili je ona neobvezna. Strukture podataka mogu se mijenjati bez prethodnog određivanja tablica, što podupire agilni razvoj i brze izmjene: nova polja i ugniježđeni objekti mogu se pohranjivati bez migracije baze.*
- *Horizontalna skalabilnost: umjesto povećavanja snage jednog računala, NoSQL sustavi omogućuju raspodjelu podataka na više poslužitelja ili čvorova. Takav pristup omogućuje rast performansi dodavanjem čvorova.*
- *Visoka dostupnost i otpornost na kvarove: većina NoSQL baza replicira podatke na više čvorova, često uz distribuirane algoritme konsenzusa ili replikaciju temeljenu na kvorumu. Sustav tako može nastaviti raditi i kada pojedini čvorovi otkažu.*

- *Modeli slabije dosljednosti: u distribuiranim sustavima snažna dosljednost među čvorovima može povećati latenciju i uzrokovati nedostupnost. Mnogi NoSQL sustavi primjenjuju model BASE (Basically Available, Soft State, Eventually Consistent), koji ublažava zahtjeve modela ACID radi bolje skalabilnosti i otpornosti na kvarove.*
- *Optimizacija za određene primjene: umjesto jednog relacijskog modela za sve potrebe, NoSQL tehnologije prilagođene su pojedinim vrstama podataka i obrascima pristupa. Neke su prikladne za predmemoriranje, druge za dokumente, obilazak grafova ili analitičku obradu.*
- *Distribuirana arhitektura: NoSQL sustavi projektiraju se kao skupine poslužitelja koje dijele opterećenje. Particioniranje (sharding) raspodjeljuje podatke, a replikacija povećava trajnost pohrane i otpornost na kvarove.*

Jedno od najvažnijih teorijskih polazišta NoSQL baza jest CAP teorem Erica Brewera. Prema njemu distribuirani podatkovni sustav ne može istodobno jamčiti sva tri sljedeća svojstva:

- *Dosljednost (Consistency): svi čvorovi u istom trenutku raspolažu istim podacima.*
- *Dostupnost (Availability): na svaki zahtjev stiže odgovor, čak i ako sadržava zastarjele podatke.*
- *Tolerancija na mrežno particioniranje (Partition Tolerance): sustav nastavlja raditi unatoč prekidima veze između dijelova mreže ili drugim mrežnim kvarovima.*

U slučaju mrežnog particioniranja potrebno je odrediti prioritete, što dovodi do različitih kompromisa. Sustavi CA, rijetki u distribuiranim okruženjima, naglašavaju dosljednost i dostupnost. Sustavi CP (npr. HBase ili MongoDB u određenim konfiguracijama) daju prednost dosljednosti i toleranciji particioniranja. Sustavi AP (npr. Cassandra i CouchDB) naglašavaju dostupnost i toleranciju particioniranja uz mogućnost naknadnog usklađivanja podataka.

Radi skalabilnosti i otpornosti mnoge NoSQL baze slijede načela modela BASE:

- ***Osnovna dostupnost (Basically Available): sustav je dostupan u većini slučajeva.***
- ***Promjenjivo stanje (Soft State): podaci se mogu mijenjati tijekom vremena i bez izravne korisničke radnje, primjerice zbog replikacije.***
- ***Naknadna dosljednost (Eventually Consistent): ako nema novih promjena, podaci će se s vremenom uskladiti na svim čvorovima.***

Taj model omogućuje veliku propusnost i otpornost na kvarove, posebno u infrastrukturi u oblaku i distribuiranim okruženjima.

2.2. Osnovni NoSQL modeli podataka

NoSQL baze uobičajeno se dijele u četiri osnovne skupine prema načinu pohrane i pristupa podacima:

1. Spremišta ključ–vrijednost
2. Dokumentne baze podataka
3. Baze sa skupinama stupaca
4. Grafovske baze podataka.

Spremišta ključ–vrijednost najjednostavniji su NoSQL model. Svaki zapis sadržava jedinstveni ključ povezan s vrijednošću, koja može biti proizvoljan binarni objekt, JSON dokument ili serijalizirana struktura. Sama baza ne tumači unutarnju strukturu vrijednosti, nego je dohvaća prema ključu. Takva jednostavnost omogućuje iznimno brze operacije čitanja i pisanja, bez spajanja tablica, složenog indeksiranja ili relacijskih ograničenja. Glavne su prednosti brzina, a ograničenje skromne mogućnosti upita.

Primjeri su Redis, Amazon DynamoDB i Riak KV. Redis je spremište podatkovnih struktura u radnoj memoriji s mogućnošću trajne pohrane i razmjene poruka po modelu objava–pretplata (pub/sub). Amazon DynamoDB potpuno je upravljana usluga u oblaku nadahnuta arhitekturom Dynamo. Riak KV distribuirana je baza ključ–vrijednost usmjerena na otpornost na kvarove.

U praćenju okoliša spremišta ključ–vrijednost osobito su korisna kada je mala latencija važnija od složenih upita. Nadzorne ploče u stvarnom vremenu, primjerice, često prikazuju najnovije očitanje pojedinog senzora. Umjesto pretraživanja čitavog vremenskog niza, posljednja se vrijednost pohranjuje pod unaprijed određenim ključem.

Razmotrimo sljedeći primjer u Redisu:

```
SET sensor_1001_last_reading '{"temperature": 12.6, "humidity": 78.5}'
```

U ovoj naredbi ključ sensor_1001_last_reading jednoznačno određuje spremljenu vrijednost: JSON niz s najnovijim mjerenjima okoliša. Kada nadzorna ploča zatraži posljednje očitanje, izvršava se naredba:

```
GET sensor_1001_last_reading
```

Baza odmah vraća pohranjenu vrijednost bez filtriranja i agregacije. Učinkovitost proizlazi iz Redisove pohrane u radnoj memoriji i indeksiranja temeljenog na raspršivanju, čime se omogućuje dohvat u konstantnom vremenu.

Sustavi ključ–vrijednost koriste se i za brojače, primjerice za bilježenje broja opažanja vrsta unutar kratkog vremenskog razdoblja:

```
INCR site_001_detection_counter
```

Naredba atomarno povećava pripadajući brojač. Atomarnost osigurava ispravnost i kada više procesa istodobno mijenja njegovu vrijednost, što je važno u sustavima s učestalim ekološkim događajima.

Spremišta ključ–vrijednost vrlo su brza i jednostavna, ali imaju ograničene analitičke mogućnosti. Primjerice, bez dodatnih indeksa ne mogu učinkovito izvršiti upit „Pronađi sve senzore s temperaturom višom od 30 °C", jer ne poznaju unutarnju strukturu vrijednosti. Stoga se najčešće koriste kao dijelovi širih arhitektura koje povezuju više vrsta baza.

Dokumentne baze podataka pohranjuju podatke u fleksibilnim polustrukturiranim formatima, kao što su JSON, BSON i XML. Svaki dokument objedinjuje informacije o entitetu, uključujući ugniježđene objekte i polja. Za razliku od modela ključ–vrijednost, dokumentne baze poznaju strukturu zapisa pa omogućuju indeksiranje i pretraživanje ugniježđenih atributa.

Tipične su primjene sustavi za upravljanje sadržajem, katalozi internetskih trgovina, korisnički profili i zapisnici događaja. MongoDB poznata je dokumentna baza s fleksibilnim shemama i bogatim mogućnostima upita. CouchDB koristi JSON i HTTP API te naglašava replikaciju i rad bez stalne internetske veze. ArangoDB višemodelna je baza koja podržava dokumente, ključ–vrijednost i grafove. Glavna su obilježja dokumentnih baza:

- podatkovni model bez obvezne sheme koji omogućuje agilnost i brze izmjene;
- bogati upitni jezici i indeksiranje, uključujući ugniježđena polja;
- mogućnost horizontalnog skaliranja particioniranjem i replikacijskim skupinama.

U okolišnim primjenama dokumentne baze pogodne su za polustrukturirana ekološka opažanja čiji se zapis razlikuje među lokacijama ili mijenja s vremenom. Jedna postaja može mjeriti temperaturu i vlažnost, a druga uz to salinitet i UV zračenje.

Razmotrimo primjer u MongoDB-u:

```
db.sensor_readings.insertOne({
    sensor_id: "sensor_1001",
    site_id: "wetland_A",
    timestamp: ISODate("2025-10-15T09:12:00Z"),
    temperature: 12.6,
    humidity: 78.5,
```

```
    soil_moisture: 0.21
});
```

Ovaj dokument predstavlja jedno opažanje stanja okoliša. MongoDB ga pohranjuje bez unaprijed definirane sheme. Ako se pojavi novi parametar, primjerice tlak zraka, može se jednostavno dodati:

```
db.sensor_readings.insertOne({
    sensor_id: "sensor_1002",
    site_id: "wetland_A",
    timestamp: ISODate("2025-10-15T09:15:00Z"),
    temperature: 13.1,
    humidity: 76.4,
    air_pressure: 1012.3
});
```

Nije potrebna promjena sheme. Ta je fleksibilnost važna u ekološkim sustavima u kojima se mjerni parametri mijenjaju zajedno s istraživačkim ciljevima.

Dokumentne baze omogućuju i indeksiranje pojedinih polja:

```
db.sensor_readings.createIndex({ site_id: 1, timestamp: -1 });
```

Složeni indeks omogućuje učinkovito dohvaćanje najnovijih mjerenja za određenu lokaciju. Prvo polje (site_id) logički grupira podatke prema lokaciji, a drugo (timestamp) omogućuje vremensko sortiranje. Primjerice, kada se izvrši upit:

```
db.sensor_readings.find({ site_id: "wetland_A" }).sort({ timestamp: -1
}).limit(10);
```

MongoDB koristi indeks za brz povrat najnovijih zapisa bez pregledavanja cijele zbirke.

Dokumentni model povezuje fleksibilnost i izražajne mogućnosti upita, zbog čega je među najraširenijim NoSQL pristupima u sustavima okoliša.

Baze sa skupinama stupaca organiziraju podatke u povezane skupine stupaca. Primjenjuju se za vremenske nizove, agregaciju zapisnika i događaja te opsežne analize i skladištenje podataka. Apache Cassandra odlikuje se visokom dostupnošću, skalabilnošću i otpornošću na kvarove. HBase je otvorena implementacija modela Bigtable izgrađena na Hadoopovu sustavu HDFS. ScyllaDB visokoučinkovita je implementacija slična Cassandri, napisana u jeziku C++. Glavna su obilježja:

- vrlo učinkovit zapis podataka i dobra skalabilnost;
- podesive razine dosljednosti;

- optimizacija za pohranu sa širokim skupovima stupaca;
- prikladnost za analitička opterećenja.

Taj je model posebno učinkovit za vremenske nizove podataka o okolišu, u kojima se mjerenja bilježe uzastopno i često pretražuju prema vremenskom rasponu.

Razmotrimo definiciju tablice u Cassandri:

```
CREATE TABLE ecology.sensor_data (
    site_id text,
    day text,
    timestamp timestamp,
    temperature float,
    humidity float,
    soil_moisture float,
    PRIMARY KEY ((site_id, day), timestamp)
) WITH CLUSTERING ORDER BY (timestamp DESC);
```

Primarni ključ sastoji se od particijskog ključa (site_id, day) i stupca za grupiranje i sortiranje (timestamp). Particijski ključ određuje raspodjelu podataka među čvorovima klastera. Sva mjerenja s iste lokacije i istog dana pohranjuju se zajedno, što omogućuje učinkovite upite nad dnevnim rasponima.

Stupac za grupiranje određuje fizički redoslijed redaka unutar particije. Budući da su vremenske oznake poredane silazno, posljednja se očitanja mogu brzo dohvatiti:

```
SELECT *
FROM ecology.sensor_data
WHERE site_id = 'wetland_A'
AND day = '2025-10-15'
LIMIT 5;
```

Upit ne pregledava cijeli skup podataka, nego izravno pristupa odgovarajućoj particiji i dohvaća najnovije zapise prema redoslijedu grupiranja.

Baze sa skupinama stupaca posebno su prikladne za intenzivan prihvat podataka. Kada tisuće senzora svake sekunde šalju očitanja, Cassandra raspodjeljuje upise među čvorovima dosljednim raspršivanjem i smanjuje uska grla. Zato odgovara i nacionalnim sustavima za praćenje okoliša.

Grafovske baze podataka prikazuju i pohranjuju podatke grafovima sastavljenima od čvorova, bridova i svojstava. Namijenjene su istraživanju odnosa i obrazaca među entitetima. Primjenjuju se u društvenim mrežama, sustavima preporuka, otkrivanju prijevara, mrežnoj analizi, grafovima znanja i semantičkim modelima.

Neo4j jedna je od najpoznatijih grafovskih baza i koristi upitni jezik Cypher. Amazon Neptune upravljana je grafovska usluga koja podržava grafove sa svojstvima i RDF grafove. OrientDB povezuje dokumentni i grafovski model. Glavna su obilježja grafovskih baza:

- optimiziran obilazak složenih odnosa;
- izražajni upitni jezici za pronalaženje putova i prepoznavanje obrazaca;
- prikladnost za povezane podatke kod kojih su odnosi jednako važni kao sami entiteti.

Grafovske baze modeliraju podatke kao čvorove povezane odnosima (bridovima), pri čemu čvorovi i odnosi mogu sadržavati svojstva. To prirodno odgovara ekološkim sustavima, u kojima su međudjelovanja entiteta često jednako važna kao i sami entiteti.

Primjerice, vrste nastanjuju pojedine lokacije, lokacije pripadaju regijama, a regije mogu biti dio zaštićenih mreža. U relacijskom modelu takvi bi odnosi zahtijevali brojne operacije spajanja, čija računalna zahtjevnost raste s dubinom povezivanja.

U Neo4j-u te se veze mogu izričito modelirati:

```
MERGE (s:Species {name: "Anas platyrhynchos"})
MERGE (site:Site {id: "wetland_A"})
MERGE (region:Region {name: "Northern Basin"})
MERGE (s)-[:OBSERVED_AT]->(site)
MERGE (site)-[:LOCATED_IN]->(region);
```

Svaka naredba MERGE osigurava postojanje čvora ili veze bez stvaranja duplikata. Izričiti odnosi OBSERVED_AT i LOCATED_IN omogućuju pregledne upite za obilazak grafa.

Za izračun bioraznolikosti po regiji može se izvršiti sljedeći upit:

```
MATCH (s:Species)-[:OBSERVED_AT]->(site:Site)-[:LOCATED_IN]-
>(region:Region)
RETURN region.name, COUNT(DISTINCT s) AS biodiversity
ORDER BY biodiversity DESC;
```

Upit obilazi graf umjesto relacijskih spajanja. Budući da grafovske baze izravno pohranjuju susjedne veze, obilazak je učinkovit i u složenim mrežama.

Grafovski modeli posebno su korisni za:

- modeliranje hranidbenih mreža;
- analizu migracijskih mreža;
- procjenu povezanosti staništa;
- istraživanje međudjelovanja ekosustava.

Njihova je prednost analiza odnosa, a ne masovna numerička agregacija.

2.3. Usporedba NoSQL i relacijskih baza podataka

Za razliku od relacijskih baza sa strogim shemama i normalizacijom, NoSQL baze potiču denormalizirani pristup prilagođen aplikaciji. Modeliranje snažno ovisi o načinu na koji aplikacija pristupa podacima. Glavna načela obuhvaćaju:

- *Projektiranje prema upitima, a ne samo prema integritetu podataka: model treba odražavati najčešće načine čitanja i zapisivanja podataka.*
- *Prihvaćanje denormalizacije: ponavljanjem pojedinih podataka smanjuje se potreba za složenim spajanjima i povećava učinkovitost.*
- *Korištenje agregata: povezani podaci (npr. svi detalji narudžbe) objedinjuju se u jedan dokument ili skupinu stupaca.*
- *Ugrađivanje ili referenciranje: bira se između ugradnje povezanih podataka (brže čitanje) i upućivanja na njih (bolja normalizacija).*

Dok SQL pruža jedinstven upitni jezik, NoSQL sustavi koriste različite mehanizme:

1. dohvat prema ključu u spremištima ključ–vrijednost;
2. upite nalik JSON-u u dokumentnim bazama;
3. CQL (Cassandra Query Language) u bazama sa skupinama stupaca;
4. jezike za obilazak grafa poput Cyphera i Gremlina u grafovskim bazama.

Radi učinkovitijeg izvršavanja upita primjenjuju se indeksne strukture kao što su raspršeni indeksi, indeksi B-stabla i indeksi za pretraživanje cijelog teksta. Neki sustavi, primjerice MongoDB, podržavaju složene, prostorne i tekstne indekse, dok se Cassandra oslanja na particijske ključeve i stupce za grupiranje. Sve češći je pristup poliglotske pohrane, odnosno primjene više tehnologija baza podataka u jednoj aplikaciji: Redis za predmemoriju i sesije, MongoDB za profile i sadržaj, Cassandra za velike tokove događaja, a Neo4j za analizu odnosa. Povezivanje se ostvaruje API-jima, podatkovnim cjevovodima ili redovima poruka. Pružatelji oblaka poput AWS-a i Azurea nude upravljene NoSQL usluge, primjerice DynamoDB i Cosmos DB. Unatoč prednostima, NoSQL donosi i izazove:

- složeno upravljanje dosljednošću distribuiranih sustava;
- kompromise pri modeliranju koji zahtijevaju dobro poznavanje obrazaca pristupa;
- nedostatak standardizacije upitnih jezika i API-ja;
- sigurnost i upravljanje transakcijama koji se razlikuju od relacijskih sustava.

Novi tehnološki pristupi nastoje ukloniti navedene nedostatke. NewSQL baze, poput CockroachDB-a i Google Spannera, povezuju NoSQL skalabilnost i transakcije ACID. Višemodelne baze, primjerice ArangoDB i OrientDB, podržavaju dokumente, grafove i ključ–vrijednost unutar jednog sustava. Budućnost nerelacijskog upravljanja podacima povezana je s hibridnim arhitekturama koje se prilagođavaju opterećenjima i uključuju optimizaciju upita umjetnom inteligencijom, automatsko particioniranje i interoperabilnost modela. NoSQL i nerelacijske baze promijenile su upravljanje podacima u distribuiranim sustavima visokih performansi. Omogućuju fleksibilnost, skalabilnost i prilagodbu različitim vrstama podataka: od predmemorija i dokumenata do stupčane analitike i zaključivanja nad grafovima. Razumijevanje njihovih prednosti, kompromisa i primjena važno je za suvremene softverske inženjere i arhitekte podataka.

2.4. NoSQL baze u kontekstu okoliša

Tradicionalno relacijsko projektiranje polazi od definiranja tablica, odnosa i ograničenja, prije implementacije aplikacijske logike. NoSQL sustavi, nasuprot tomu, modeliraju podatke prema potrebama aplikacije. Arhitekti najprije analiziraju kako će se podacima pristupati, kako će se pretraživati i kako će sustav rasti. To je osobito važno za podatke o okolišu, gdje su upiti usmjereni na nedavna senzorska mjerenja, određena prostorna područja ili agregirane pokazatelje ekosustava. Takvi obrasci pristupa zahtijevaju fleksibilnu strukturu i raspodjelu.

Razmotrimo jednostavnu relacijsku shemu za praćenje okoliša:

```
CREATE TABLE sensor_readings (
    sensor_id VARCHAR(50),
    measurement_time TIMESTAMP,
    temperature FLOAT,
    humidity FLOAT,
    soil_moisture FLOAT,
    PRIMARY KEY (sensor_id, measurement_time)
);
```

Iako se shema na prvi pogled čini dostatnom, problemi nastaju pri uvođenju novih varijabli okoliša. Dodavanje atributa zahtijeva izmjenu sheme:

```
ALTER TABLE sensor_readings
ADD air_pressure FLOAT;
```

U infrastrukturi koja se brzo razvija i u kojoj se senzori često nadograđuju, ponovljene promjene sheme mogu otežati rad i povećati troškove.

Dokumentna baza kao što je MongoDB, za razliku od toga, omogućuje pohranu istog zapisa bez unaprijed utvrđene sheme:

```
db.sensor_readings.insertOne({
    sensor_id: "sensor_1001",
    timestamp: ISODate("2025-10-15T09:12:00Z"),
    temperature: 12.6,
    humidity: 78.5,
    soil_moisture: 0.21,
    air_pressure: 1012.3
});
```

Struktura je fleksibilna: novi se parametar može uključiti u buduće dokumente bez izmjene postojećih zapisa. Upravo je to jedna od temeljnih prednosti NoSQL-a u dinamičnim okolišnim sustavima.

Osim fleksibilnosti sheme, važna je razlika distribuirana skalabilnost. Platforme za praćenje okoliša prikupljaju podatke od stotina ili tisuća geografski raspoređenih senzora, pri čemu centralizirana baza može postati usko grlo. Distribuirani NoSQL sustavi problem rješavaju particioniranjem podataka na više čvorova. U Cassandri se raspodjela, primjerice, temelji na particijskim ključevima:

```
CREATE TABLE sensor_data (
    site_id text,
    day text,
    timestamp timestamp,
    temperature float,
    humidity float,
    PRIMARY KEY ((site_id, day), timestamp)
) WITH CLUSTERING ORDER BY (timestamp DESC);
```

Složeni particijski ključ (site_id, day) okuplja dnevna mjerenja pojedine lokacije, uz ravnomjernu raspodjelu među čvorovima klastera. Dodavanjem novih lokacija i čvorova kapacitet se može horizontalno povećavati bez preoblikovanja sheme.

Distribuirani sustavi ipak zahtijevaju kompromise. CAP teorem pokazuje da pri mrežnom particioniranju nije moguće istodobno jamčiti strogu dosljednost i potpunu dostupnost. Mnogi sustavi za praćenje okoliša daju prednost dostupnosti, osobito u hitnim situacijama. Primjerice, replikacijski skupovi MongoDB-a konfiguriraju se radi redundantnosti:

```
rs.initiate({
   _id: "replicaSet1",
   members: [
      { _id: 0, host: "node1:27017" },
```

```
      { _id: 1, host: "node2:27017" },
      { _id: 2, host: "node3:27017" }
   ]
});
```

Ako primarni čvor otkaže, u toj konfiguraciji jedan od sekundarnih čvorova automatski preuzima njegovu ulogu i sustav može nastaviti raditi. Tijekom prebacivanja moguće su privremene nedosljednosti, no dostupnost ostaje posebno važna za praćenje ekoloških uvjeta u stvarnom vremenu.

Pojedini NoSQL modeli prilagođeni su različitim vrstama podataka o okolišu. Dokumentne baze obrađuju polustrukturirana ekološka opažanja, spremišta ključ–vrijednost pružaju brzo predmemoriranje za nadzorne ploče, a baze sa skupinama stupaca omogućuju intenzivan prihvat vremenskih nizova. Grafovske baze posebno su korisne za ekološke odnose; primjerice, u Neo4j-u vrste i staništa mogu se izravno povezati:

```
MERGE (s:Species {name: "Anas platyrhynchos"})
MERGE (site:Site {id: "site_001"})
MERGE (s)-[:OBSERVED_AT]->(site);
```

Takav prikaz omogućuje složene upite nad povezanim staništima i vrstama koje su ondje zabilježene:

```
MATCH (s:Species)-[:OBSERVED_AT]->(site:Site)
RETURN site.id, COUNT(DISTINCT s) AS biodiversity
ORDER BY biodiversity DESC;
```

Kada se povećava dubina odnosa, takve je grafovske operacije teško jednako učinkovito izvoditi u tradicionalnim relacijskim sustavima.

Suvremene platforme za okoliš u praksi se rijetko oslanjaju na samo jednu vrstu baze. Umjesto toga primjenjuju poliglotsku pohranu: više namjenskih sustava povezuju unutar jedinstvene arhitekture. Tako Kafka može prihvaćati tokove podataka, Cassandra pohranjivati senzorska mjerenja, MongoDB čuvati metapodatke, a Neo4j analizirati odnose. Sastavnice se povezuju API-jima i tokovima podataka, a ne velikim relacijskim spajanjima.

Takva strategija omogućuje svakom podatkovnom modelu da iskoristi svoje prednosti uz očuvanje usklađenosti cjeline. Raspodjelom zadaća među specijaliziranim sustavima povećavaju se skalabilnost, otpornost na kvarove i prilagodljivost promjenjivim zahtjevima praćenja okoliša.

Osnove NoSQL-a i distribuiranih podatkovnih arhitektura odražavaju širi razvoj upravljanja podacima o okolišu. Umjesto centraliziranih i krutih shema uvode se distribuirani, skalabilni i

fleksibilni sustavi prikladni za učestala senzorska mjerenja i raznorodne ekološke podatke. Poznavanje tih načela ključno je za izgradnju pouzdanih platformi za praćenje u stvarnom vremenu, dugoročne analize i donošenje održivih odluka o okolišu.

3. Studija slučaja: platforma za praćenje bioraznolikosti i okoliša u stvarnom vremenu

3.1. Konceptualni okvir i ciljevi

Ova studija slučaja pokazuje kako distribuirana NoSQL arhitektura može omogućiti skalabilnu pohranu podataka o okolišu i njihovo sustavno ekološko tumačenje. Cilj je projektirati i implementirati platformu za praćenje bioraznolikosti i okoliša u stvarnom vremenu, koja raznorodna opažanja pretvara u prepoznatljiva ekološka stanja.

Sustav povezuje učestala senzorska mjerenja okoliša, meteorološke vremenske nizove i zapise o pojavljivanju vrsta. Izvori se razlikuju prema strukturi, vremenskoj razlučivosti i značenju. Glavni je izazov uskladiti ih u jedinstvenom analitičkom postupku uz očuvanje skalabilnosti, fleksibilnosti i razumljivosti rezultata.

Upoznajmo logiku analitičkog postupka koji obuhvaća distribuirani prihvat i pohranu, izradu značajki te nenadzirano prepoznavanje ekoloških režima. Krajnji je cilj, na temelju objedinjenih pokazatelja okoliša i bioraznolikosti, prepoznati stabilna, prijelazna i kritična stanja ekosustava.

3.2. Podatkovne domene i strukturna obilježja

Platforma za praćenje objedinjuje tri glavne podatkovne domene.

Prvu domenu čine IoT mjerenja okoliša. Vlažnost tla, površinska temperatura i oborine bilježe se u minutnim intervalima na više močvarnih lokacija. Riječ je o opsežnim vremenskim nizovima sa strogo zadanim vremenskim redoslijedom.

Druga domena obuhvaća meteorološke podatke koji se svaki sat dohvaćaju iz vanjskog API-ja. Oni opisuju atmosferske uvjete koji utječu na obrasce hidrološkog opterećenja.

Treća domena uključuje zapise o pojavljivanju vrsta iz repozitorija opažanja. To su polustrukturirani dokumenti s identifikatorima vrsta, geografskim koordinatama, vremenskim oznakama i pripadajućim metapodacima.

Svaka domena ima posebna strukturna obilježja i učestalost ažuriranja, prikazane u tablici 1.

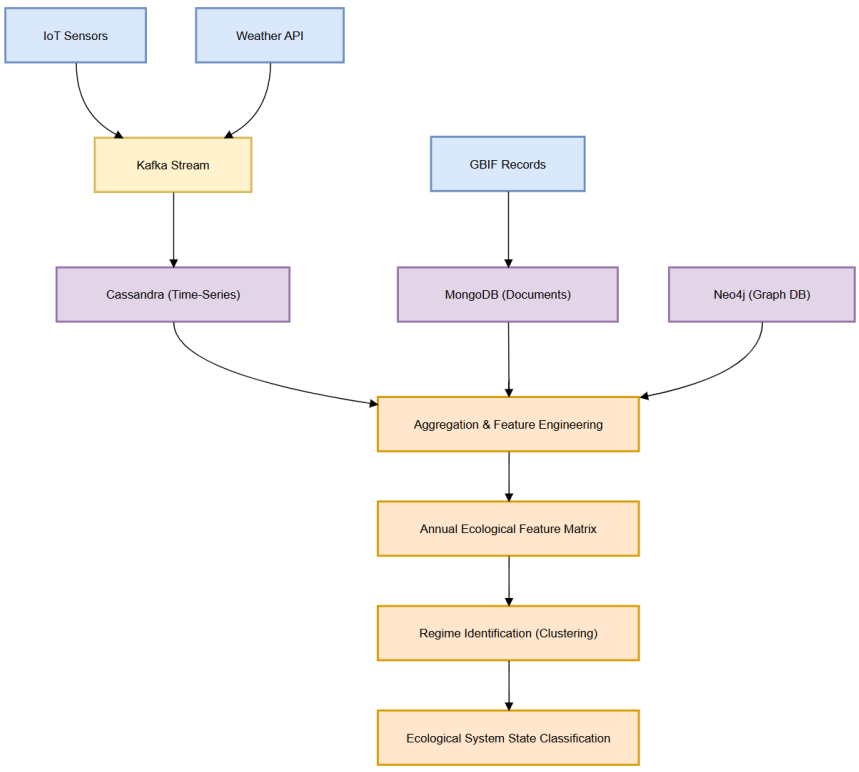
Tablica 1. Strukturna obilježja i ekološke uloge povezanih podatkovnih domena

Podatkovna domena	**Struktura podataka**	**Vremenska razlučivost**	**Ekološka uloga**
IoT senzori okoliša	Vremenski nizovi	Minutna	Lokalno hidrološko i toplinsko opterećenje
Meteorološki API	Vremenski nizovi	Satna	Klimatski utjecaji
Zapisi o bioraznolikosti	Polustrukturirani dokumenti	Temeljena na događajima	Biološki odgovor
Odnosi vrsta i lokacija	Grafovska struktura	Potaknuta događajima	Ekološka povezanost

Zbog te raznorodnosti nužna je strategija poliglotske pohrane, u kojoj se odabiru različiti NoSQL modeli prema strukturi podataka i potrebama analize.

3.3. Distribuirana arhitektura i tok podataka

Razmotrimo distribuiranu arhitekturu i cjelokupan tok podataka prikazan na slici 1.



Slika 1. Arhitektura platforme i tok podataka

Arhitektura odvaja prihvat podataka, namjensku pohranu, integraciju i analitičko modeliranje u zasebne, međusobno povezane slojeve.

IoT i meteorološki tokovi najprije prolaze kroz međuspremnički sloj zasnovan na Kafki. Time se proizvođači podataka odvajaju od sustava pohrane, a privremenim zadržavanjem podataka tijekom prekida povećava otpornost sustava.

Vremenski nizovi mjerenja okoliša pohranjuju se u Cassandri. Njezina arhitektura zasnovana na particijama podržava horizontalno skaliranje i velik kapacitet zapisivanja pa je prikladna za kontinuirani prihvat minutnih mjerenja.

Zapisi o pojavljivanju vrsta pohranjuju se u MongoDB-u. Dokumentni model omogućuje fleksibilan prikaz promjenjivih atributa vrsta i podržava prostorno indeksiranje.

Ekološki odnosi među vrstama, lokacijama i regijama modeliraju se u Neo4j-u. Grafovski prikaz omogućuje učinkovit obilazak veza koji otkriva obrasce povezanosti ekosustava.

Takva slojevita arhitektura omogućuje svakom sustavu pohrane da obavlja zadatke za koje je najprikladniji i pritom doprinosi zajedničkom analitičkom rezultatu.

3.4. Projektiranje pohrane i agregacija pokazatelja

Kao što je objašnjeno, vremenski nizovi okolišnih mjerenja pohranjuju se u Cassandri. Za to se može upotrijebiti složeni particijski ključ:

```
CREATE TABLE ecology.environmental_readings (
  site_id text,
  day text,
  timestamp timestamp,
  temperature float,
  precipitation float,
  soil_moisture float,
  PRIMARY KEY ((site_id, day), timestamp)
) WITH CLUSTERING ORDER BY (timestamp DESC);
```

Particijski ključ (site_id, day) raspodjeljuje podatke među čvorovima i okuplja dnevna mjerenja svake lokacije. Stupac timestamp određuje njihov vremenski redoslijed unutar particije.

Za izračun godišnjih pokazatelja okoliša agregacijski upiti sažimaju minutna mjerenja u strukturirane varijable opterećenja:

```
SELECT
  site_id,
  EXTRACT(YEAR FROM timestamp) AS year,
  AVG(temperature) AS avg_temp_year,
```

```
  COUNT(CASE WHEN precipitation < 1 THEN 1 END) AS dry_days_year,
  COUNT(CASE WHEN soil_moisture < 0.20 THEN 1 END) AS drought_events
FROM ecology.environmental_readings
GROUP BY site_id, EXTRACT(YEAR FROM timestamp);
```

Ti pokazatelji opisuju toplinsko opterećenje, manjak oborina i učestalost hidrološkog stresa.

Zapisi o pojavljivanju vrsta pohranjuju se u MongoDB-u kao fleksibilni JSON dokumenti:

```
db.species_occurrences.insertOne({
  species: "Rana temporaria",
  site_id: "wetland_A",
  eventDate: ISODate("2023-05-14"),
  coordinates: {
    type: "Point",
    coordinates: [30.5234, 50.4501]
  }
});
```

Broj različitih vrsta na pojedinoj lokaciji u određenoj godini izračunava se agregacijskim postupkom:

```
db.species_occurrences.aggregate([
  {
    $group: {
      _id: { site_id: "$site_id", year: { $year: "$eventDate" }},
      species_set: { $addToSet: "$species" }
    }
  },
  {
    $project: {
      site_id: "$_id.site_id",
      year: "$_id.year",
      species_richness: { $size: "$species_set" }
    }
  }
])
```

Na taj način dobivaju se pokazatelji odgovora bioraznolikosti usklađeni s varijablama opterećenja okoliša.

3.5. Izrada godišnje matrice ekoloških značajki

Agregirani pokazatelji okoliša i bioraznolikosti spajaju se u jedinstveni analitički prikaz: godišnju matricu ekoloških značajki (tablica 2). Svaki redak prikazuje opažanje za jednu lokaciju i godinu, a stupci predstavljaju izvedena ekološka obilježja.

Tablica 2. Godišnja matrica ekoloških značajki

Lokacija	Godina	Prosj. temperatura	Sušni dani	Sušni događaji	Bogatstvo vrsta
Močvara A	2018	14.2	210	2	58
Močvara A	2019	15.6	242	5	47
Močvara A	2020	17.1	275	9	32

Matrica povezuje opterećenje okoliša i odgovor bioraznolikosti unutar jedne jedinice opažanja. Prije grupiranja značajke se standardiziraju kako bi njihov doprinos bio uravnotežen:

```
from pyspark.ml.feature import StandardScaler

scaler = StandardScaler(
    inputCol="raw_features",
    outputCol="scaled_features",
    withStd=True,
    withMean=True
)
```

Standardizacijom se sprječava da pojedina varijabla prevlada u postupku grupiranja samo zbog razlike u mjernoj skali.

3.6. Prepoznavanje i analiza struktura ekoloških režima

Na standardiziranu matricu značajki primjenjuje se nenadzirano grupiranje radi prepoznavanja različitih ekoloških režima.

```
kmeans = KMeans(k=3, seed=1)
model = kmeans.fit(data)
predictions = model.transform(data)
```

Centroidi klastera upućuju na tri ekološka režima (tablica 3).

Tablica 3. Ekološki režimi

Klaster	Prosj. temperatura	Sušni događaji	Bogatstvo vrsta	Tumačenje režima
1	Umjerena	Niska	Visoka	Stabilan
2	Povišena	Srednja	Umjerena	Prijelazan

3	Visoka	Česti	Niska	Kritično opterećenje

Vremenska raspodjela klastera pokazuje prijelaze među režimima (tablica 4).

Tablica 4. Klasteri po godinama

Godina	Klaster	Godina	Klaster	Godina	Klaster
2015	1	2018	2	2021	3
2016	1	2019	3	2022	3
2017	2	2020	3	2023	3

Matrica prijelaza potvrđuje ovisnost o prethodnim stanjima (tablica 5).

Tablica 5. Matrica prijelaza

Iz \ u	Klaster 1	Klaster 2	Klaster 3
Klaster 1	1	1	0
Klaster 2	0	1	1
Klaster 3	0	0	4

Kada ekosustav prijeđe u kritični režim, vjerojatnost njegova zadržavanja raste, što može upućivati na smanjenu otpornost. Ta je informacija važna za odlučivanje u upravljanju ekosustavom.

Studija pokazuje da NoSQL sustavi nisu samo skalabilna rješenja za pohranu. U kombinaciji sa strukturiranom agregacijom i oblikovanjem značajki omogućuju višedimenzionalno ekološko modeliranje. Povezivanje pokazatelja opterećenja iz vremenskih nizova i pokazatelja bioraznolikosti otkriva ekološke režime koji odražavaju trajne promjene u okolišu.

Namjenskom distribuiranom pohranom, integracijom značajki i prepoznavanjem režima platforma pretvara izvorne podatke o okolišu u razumljiva stanja sustava. Takve informacije mogu poduprijeti upravljanje ekosustavima, procjenu rizika i rano upozoravanje.

Praktične vježbe

Zadatak 1

1. Sastavite heterogeni skup podataka o okolišu i ekologiji koristeći izvore iz tablice u nastavku.

Skup podataka	Opis	Poveznica
GBIF (Global Biodiversity Information Facility)	Veliki repozitorij opažanja vrsta i podataka o bioraznolikosti	https://www.gbif.org/
Europska agencija za okoliš – DataHub	Skupovi podataka o okolišu i ekologiji iz cijele Europe	https://www.eea.europa.eu/en/datahub
UKCEH (UK Centre for Ecology & Hydrology)	Environmental monitoring datasets; time-series and spatial metadata	https://www.ceh.ac.uk/data
NEON (National Ecological Observatory Network)	Otvoreni izvori ekoloških podataka	https://libguides.cmich.edu/lifesciencedata/ecological

2. Analizirajte obilježja sastavljenog skupa podataka.

3. Predložite arhitekturu NoSQL pohrane za rad s tim podacima.

Rezultat: izvješće o dobivenim rezultatima.

Kriteriji vrednovanja: potpunost rezultata i jasnoća tumačenja.

Zadatak 2

Zamislite da surađujete s agencijom za zaštitu okoliša koja prati pojavljivanje vrsta, obilježja staništa (temperaturu, relativnu vlažnost, vlažnost tla) i opažanja događaja (npr. aktiviranje fotoklopki ili senzorskih alarma). Predložite skup NoSQL tehnologija za izgradnju programskog sustava koji prikuplja tokove podataka sa senzora i fotoklopki na brojnim lokacijama, podržava fleksibilne upite i omogućuje analizu ekoloških parametara.

Rezultat: izvješće o dobivenim rezultatima.

Kriteriji vrednovanja: potpunost rezultata i jasnoća tumačenja.

Provjera znanja

1. Riješite križaljku.

Prema CAP teoremu, distribuirani podatkovni sustav ne može istodobno jamčiti sva tri sljedeća svojstva:

C																		
A																		
P									—									

2. Odgovorite na pitanja višestrukog izbora.

1. Koji je glavni razlog korištenja NoSQL baza podataka?

A. Strogo provođenje ograničenja relacijskog integriteta

B. Učinkovita obrada velikih količina nestrukturiranih ili polustrukturiranih podataka

C. Pojednostavnjenje projektiranja relacijskih shema

D. Uklanjanje potrebe za sustavima upravljanja bazama podataka

2. Što od navedenoga nije vrsta NoSQL baze podataka?

A. Spremište ključ–vrijednost

B. Dokumentna baza

C. Baza sa skupinama stupaca

D. Relacijska tablična baza

3. U dokumentnoj NoSQL bazi podaci se najčešće pohranjuju kao:

A. CSV datoteke

B. dokumenti nalik JSON-u

C. zapisi fiksne duljine

D. tablice sa stranim ključevima

4. Koja je od navedenih NoSQL baza primjer spremišta ključ–vrijednost?

A. MongoDB

B. Cassandra

C. Redis

D. Neo4j

5. Koji je NoSQL model najprikladniji za mrežne podatke i odnose, primjerice društvene grafove ili odnose među vrstama u ekosustavu?

A. Baza sa skupinama stupaca

B. Spremište ključ–vrijednost

C. Grafovska baza

D. Dokumentna baza

6. Koja je glavna prednost horizontalnog skaliranja u NoSQL sustavima?

A. Smanjenje redundancije normalizacijom svih tablica

B. Povećavanje kapaciteta dodavanjem čvorova umjesto zamjenom sklopovske opreme snažnijom

C. Jamstvo potpune usklađenosti s ACID načelima

D. Automatsko pretvaranje nestrukturiranih podataka u relacijski oblik

7. Koja je sastavnica NoSQL sustava zadužena za prihvat neprekidnih tokova podataka sa senzora i IoT uređaja?

A. Mehanizam indeksiranja

B. Procesor upita

C. Sloj za prihvat podataka i obradu tokova

D. Nadzorna ploča za izvještavanje

8. Za što je baza vremenskih nizova posebno korisna u ekološkom praćenju?

A. Upravljanje podacima sa složenim odnosima

B. Pohranu uzastopnih očitanja senzora i vremenski označenih mjerenja okoliša

C. Pohranu nestrukturiranih tekstova

D. Pretraživanje cijelog teksta dokumenata

9. Koja se tehnologija često koristi za prijenos podatkovnih tokova i obradu u stvarnom vremenu u NoSQL arhitekturama?

A. Apache Kafka

B. Microsoft Access

C. Oracle RDBMS

D. SQLite

10. Koja tvrdnja najbolje opisuje dokumentnu bazu poput MongoDB-a?

A. Zahtijeva unaprijed definirane sheme i spajanje tablica.

B. Pohranjuje i dohvaća podatke u fleksibilnim, ugniježđenim dokumentima.

C. Optimizirana je za složene relacijske upite u SQL-u.

D. Ne podržava hijerarhijske i ugniježđene strukture podataka.

Literatura i dodatni izvori

1. Chodorow, K. (2019). MongoDB: The Definitive Guide (3rd ed.). O'Reilly Media.
2. Redmond, E., & Wilson, J. R. (2012). Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. Pragmatic Bookshelf.
3. Han, J., Haihong, E., Le, G., & Du, J. (2011). Survey on NoSQL database. Proceedings of the 2011 6th International Conference on Pervasive Computing and Applications (ICPCA), 363–366. IEEE.
4. Gajendran, S., & Cook, J. (2021). Learning NoSQL: A Hands-On Guide to Big Data Management Systems. Packt Publishing.
5. Strauch, C. (2011). NoSQL Databases. Stuttgart Media University.
6. Sadalage, P. J., & Fowler, M. (2012). NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley Professional.
7. Tanenbaum, A. S., & van Steen, M. (2017). Distributed Systems: Principles and Paradigms (2nd ed.). Pearson.