Co-funded by the European Union

NoSQL in upravljanje nerelacijskih podatkov

Učni izidi

Po zaključku te učne enote bodo študenti sposobni:

- Razumeti temeljne pojme, razloge in arhitekturne razlike med relacijskimi in nerelacijskimi (NoSQL) sistemi podatkovnih zbirk, vključno z njihovimi prednostmi in omejitvami v sodobnih podatkovnih okoljih.
- Razvrstiti in opisati glavne kategorije podatkovnih zbirk NoSQL (shrambe ključ–vrednost, dokumentne podatkovne zbirke, shrambe družin stolpcev in grafovne podatkovne zbirke) ter za vsako vrsto prepoznati primerne primere uporabe in značilne tehnologije.
- Razumeti in ovrednotiti rešitve NoSQL znotraj programskih arhitektur, zlasti v kontekstu velepodatkov, računalništva v oblaku in analitike v realnem času, za podporo učinkovitim in prilagodljivim strategijam upravljanja podatkov.

1. Uvod: podatkovno intenzivni okoljski informacijski sistemi

1.1. Eksplozija okoljskih podatkov

V zadnjih dveh desetletjih je spremljanje okolja doživelo temeljito tehnološko preobrazbo. Sodobne ekološke sisteme danes opazujemo z gostimi omrežji senzorjev IoT, satelitskimi konstelacijami, avtonomnimi droni, podatkovnimi zbirkami biotske raznovrstnosti, podnebnimi modeli in platformami občanske znanosti. Ti sistemi neprekinjeno ustvarjajo ogromne količine heterogenih podatkov, ki opisujejo atmosferske razmere, hidrološke cikle, dinamiko rabe zemljišč, vzorce biotske raznovrstnosti in kazalnike zdravja ekosistemov.

V nasprotju s tradicionalnimi okoljskimi podatkovnimi nizi – ki se zbirajo periodično in obdelujejo brez povezave – sodobne infrastrukture za spremljanje ustvarjajo podatkovne tokove z veliko hitrostjo, obsegom in raznovrstnostjo. Posamezno regionalno omrežje za spremljanje kakovosti zraka lahko mesečno ustvari milijone časovno označenih odčitkov. Platforme za satelitske posnetke vsak dan zagotavljajo terabajte rastrskih podatkov. Portali biotske raznovrstnosti, kot je GBIF, hranijo več sto milijonov zapisov o pojavljanju vrst, vsak s prostorskimi in časovnimi metapodatki.

Ta preobrazba je ustvarila novo kategorijo sistemov, imenovanih podatkovno intenzivni okoljski informacijski sistemi. Ti sistemi morajo:

- povezovati heterogene strukturirane in nestrukturirane podatke,
- delovati v porazdeljenih geografskih okoljih,
- podpirati analitiko skoraj v realnem času,
- se elastično razširjati,
- in ostati odporni ob nepredvidljivih okoljskih dogodkih.

Tradicionalne podatkovne arhitekture niso bile zasnovane za takšno stopnjo dinamičnosti.

1.2. Od spremljanja okolja do porazdeljenih podatkovnih platform

Sodobne okoljske platforme delajo s podatkovnimi nizi, ki združujejo več podatkovnih modalnosti, kot prikazuje Tabela 1.

Tabela 1 – Vrste okoljskih podatkov

Vrsta podatkov	Struktura podatkov	Primeri
Časovne vrste	Zaporedni, visokofrekvenčni	Temperatura, vlažnost, PM2.5, vlažnost tal
Prostorski	Geoprostorski	Koordinate, zavarovana območja, poligoni pokrovnosti tal
Na podlagi dogodkov	Polstrukturirani	Zaznavanje prostoživečih živali, opozorila pred poplavami
Metapodatki	Dokumentni	Specifikacije senzorjev, dnevniki umerjanja
Podatki o razmerjih	Grafovno strukturirani	Povezave med vrstami in habitati, omrežja porečij
Besedilni	Nestrukturirani	Terenski zapiski, poročila, opombe

Soobstoj teh oblik podatkov prinaša arhitekturne izzive:

- toge sheme postanejo nepraktične,
- potrebno je pogosto spreminjanje shem,
- relacijske poizvedbe s številnimi operacijami JOIN z večanjem obsega poslabšujejo zmogljivost,
- navpično razširjanje postane ekonomsko neučinkovito,
- porazdeljeno podvajanje je v tradicionalnih okoljih RDBMS zapleteno.

Ker se spremljanje okolja usmerja k neprekinjenim arhitekturam v realnem času, ki so geografsko porazdeljene, postaja potreba po prilagodljivih in vodoravno razširljivih modelih podatkovnih zbirk očitna.

1.3. Zakaj relacijske arhitekture postajajo omejujoče

Sistemi za upravljanje relacijskih podatkovnih zbirk (RDBMS) se odlikujejo pri zagotavljanju transakcijske doslednosti in v okoljih z dobro opredeljenimi shemami. Vendar se srečujejo z naslednjimi omejitvami.

1. Togost sheme. Sistemi okoljskih senzorjev se nenehno razvijajo, saj se uvajajo novi merilni parametri, kot so indeks UV, slanost vode ali koncentracija mikroplastike. Zato je treba sheme podatkovnih zbirk pogosto spreminjati, ponavljajoče se operacije ALTER TABLE pa vse bolj motijo stabilnost in zmogljivost sistema.

2. Poizvedbe s številnimi združevanji. Večrazsežne okoljske poizvedbe pogosto zahtevajo združevanje podatkov, kot so podatki senzorjev, metapodatki lokacij, prostorski sloji, zapisi o vrstah itd. Zato operacije JOIN nad milijoni vrstic znatno zmanjšajo zmogljivost.

3. Omejitve vodoravnega razširjanja. Razširjanje relacijskih sistemov je tradicionalno zahtevalo navpično razširjanje (zmogljivejšo strojno opremo), ki je drago in sčasoma doseže svoje meje.

4. Neučinkovitost pri geoprostorskih in grafovnih podatkih. Čeprav obstajajo relacijske razširitve (npr. PostGIS), povečujejo kompleksnost in se v porazdeljenih arhitekturah ne razširjajo vedno učinkovito.

5. Izzivi pri sprejemu pretočnih podatkov. Visokofrekvenčno zapisovanje pretočnih podatkov lahko preobremeni transakcijske mehanizme zaklepanja.

Te omejitve ne pomenijo, da so relacijske podatkovne zbirke zastarele. Kažejo pa, da sistemi okoljskih velepodatkov potrebujejo dopolnilne ali alternativne arhitekturne paradigme.

Prav tu postane bistvenega pomena upravljanje podatkov z NoSQL in nerelacijskimi pristopi.

Sistemi NoSQL pomenijo premik od zasnove, ki izhaja iz sheme, k modeliranju, ki ga usmerja aplikacija. Namesto vprašanja: »Kako naj bodo podatki normalizirani?« se vprašamo: »Kako bomo dostopali do podatkov, po njih poizvedovali in sistem prilagajali rasti njihovega obsega?« Ta premik uvaja več temeljnih načel:

- Prilagodljive ali dinamične sheme
- Denormalizacija kot strategija za izboljšanje zmogljivosti
- Vodoravno razširjanje kot privzeta predpostavka
- Kompromisi pri porazdeljeni doslednosti
- Načrtno particioniranje in podvajanje podatkov

V okoljskih sistemih, kjer so poizvedbe pogosto usmerjene v nedavna časovna okna, prostorske regije ali kategorije ekosistemov, postane načrtovanje glede na vzorce dostopa pomembnejše od stroge normalizacije.

2. Temelji NoSQL in porazdeljenih podatkovnih arhitektur

2.1. Teoretično ozadje

Pojav tehnologij NoSQL odraža temeljit premik v načinu pojmovanja in načrtovanja porazdeljenih podatkovnih sistemov. Medtem ko so bile relacijske podatkovne zbirke zasnovane okoli vnaprej določenih shem in strogih načel normalizacije, sodobni okoljski informacijski sistemi vse pogosteje delujejo v dinamičnih, heterogenih in geografsko porazdeljenih okoljih, kjer taka togost postane omejujoča. V teh okoliščinah podatkovna zbirka ni več le strukturiran repozitorij, temveč prilagodljiva sestavina širše porazdeljene arhitekture.

Podatkovne zbirke NoSQL imajo kljub svoji raznovrstnosti več skupnih opredeljujočih značilnosti:

- *Prilagodljivost sheme*: sistemi NoSQL so običajno brez sheme ali imajo neobvezno shemo, zato se lahko podatkovne strukture razvijajo brez vnaprej določenih definicij tabel. Ta prilagodljivost podpira agilen razvoj in hitro iteriranje – razvijalci lahko shranijo nova polja ali ugnezdene objekte brez migracije podatkovne zbirke.
- *Vodoravna razširljivost*: namesto navpičnega razširjanja (dodajanja zmogljivosti enemu računalniku) so sistemi NoSQL zasnovani za *vodoravno razširjanje*, torej porazdelitev

podatkov med več strežnikov ali vozlišč. Ta zasnova zagotavlja, da lahko zmogljivost raste linearno s številom vozlišč.

- *Visoka razpoložljivost in odpornost proti napakam*: večina podatkovnih zbirk NoSQL podvaja podatke med več vozlišči, pogosto z uporabo algoritmov porazdeljenega soglasja ali podvajanja na podlagi kvoruma. To omogoča neprekinjeno delovanje tudi ob odpovedi nekaterih vozlišč.
- *Ohlapnejši modeli doslednosti*: v porazdeljenih okoljih lahko zagotavljanje stroge doslednosti med vozlišči povzroči zakasnitve in izpade. Številni sistemi NoSQL uporabljajo model *BASE* (Basically Available, Soft state, Eventually consistent), ki omili načela ACID ter tako izboljša razširljivost in odpornost proti napakam.
- *Optimizacija za določene primere uporabe*: namesto univerzalnega relacijskega modela so tehnologije NoSQL optimizirane za določene vrste podatkov in vzorce dostopa. Nekatere so idealne za predpomnjenje, druge za hrambo dokumentov, prehajanje po grafih ali analitične delovne obremenitve.
- *Porazdeljena arhitektura*: sistemi NoSQL so zasnovani kot gruče strežnikov, ki si delijo obremenitev pri obdelavi podatkov. Particioniranje podatkov (sharding) zagotavlja porazdelitev, podvajanje pa obstojnost in odpornost proti napakam.

Eden najvplivnejših teoretičnih temeljev podatkovnih zbirk NoSQL je *izrek CAP*, ki ga je predlagal Eric Brewer. Ta pravi, da v porazdeljenem podatkovnem sistemu ni mogoče hkrati zagotoviti vseh treh naslednjih lastnosti:

- *Doslednost*: vsa vozlišča vidijo iste podatke ob istem času.
- *Razpoložljivost*: vsaka zahteva prejme odgovor (tudi če je zastarel).
- Toleranca na omrežne particije**:** sistem deluje naprej kljub omrežnim particijam ali okvaram.

Sistem NoSQL mora dati prednost dvema od treh lastnosti, kar privede do kompromisov, kot so: sistemi CA (redki v porazdeljenih okoljih) poudarjajo doslednost in razpoložljivost; sistemi CP (npr. HBase, MongoDB v določenih konfiguracijah) poudarjajo doslednost in toleranco na omrežne particije; sistemi AP (npr. Cassandra, CouchDB) dajejo prednost razpoložljivosti in toleranci na omrežne particije ter dopuščajo končno doslednost.

Za podporo razširljivosti in odpornosti podatkovne zbirke NoSQL pogosto sledijo filozofiji *BASE*, ki pomeni naslednje:

- ***B****asically* ***A****vailable* pomeni, da sistem v večini primerov zagotavlja razpoložljivost.
- ***S****oft State* pomeni, da se podatki lahko sčasoma spremenijo brez uporabnikovega vnosa (zaradi podvajanja).

- ***E**ventually Consistent* pomeni, da bodo podatki med vozlišči postali dosledni, če ne bo novih posodobitev.

Ta model sistemom omogoča visoko prepustnost in odpornost proti napakam, zlasti v oblačnih in porazdeljenih infrastrukturah.

2.2. Temeljni podatkovni modeli NoSQL

Podatkovne zbirke NoSQL se običajno delijo v štiri glavne kategorije, od katerih je vsaka zasnovana za drugačne vzorce shranjevanja in potrebe po dostopu:

1. Shrambe ključ–vrednost
2. Dokumentno usmerjene podatkovne zbirke
3. Shrambe družin stolpcev
4. Grafovne podatkovne zbirke.

Shrambe ključ–vrednost predstavljajo najpreprostejšo abstrakcijo med sistemi NoSQL. V tem modelu je vsak zapis sestavljen iz enoličnega ključa, povezanega z vrednostjo, pri čemer je vrednost lahko poljuben binarni objekt, dokument JSON ali serializirana struktura. Podatkovna zbirka ne razlaga notranje strukture vrednosti, temveč jo preprosto pridobi na podlagi ključa. Ta arhitekturni minimalizem omogoča izjemno hitro branje in pisanje, saj niso vključena združevanja, kompleksno indeksiranje ali relacijske omejitve. Najpomembnejši značilnosti podatkovnih zbirk te kategorije sta izjemno hitro branje in pisanje ter omejene možnosti poizvedovanja.

Primeri shramb ključ–vrednost so *Redis*, *Amazon DynamoDB* in *Riak KV*. *Redis* je shramba podatkovnih struktur v pomnilniku, ki podpira obstojnost podatkov in sporočanje pub/sub. *Amazon DynamoDB* je v celoti upravljana storitev v oblaku, ki se zgleduje po izvirni arhitekturi Dynamo. *Riak KV* je porazdeljena podatkovna zbirka ključ–vrednost s poudarkom na odpornosti proti napakam.

V sistemih za spremljanje okolja so shrambe ključ–vrednost posebej koristne, kadar je nizka zakasnitev pomembnejša od zahtevnega poizvedovanja. Nadzorne plošče v realnem času na primer pogosto prikazujejo najnovejši odčitek senzorja za vsako napravo. Namesto pregledovanja celotnega podatkovnega niza časovne vrste lahko sistem shrani najnovejšo vrednost pod determinističnim ključem.

Oglejmo si naslednji primer z uporabo sistema Redis:

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

V tem stavku ključ sensor_1001_last_reading enolično določa shranjeno vrednost, ki je niz JSON z najnovejšimi okoljskimi meritvami. Ko nadzorna plošča za spremljanje zahteva najnovejši odčitek, izvede:

```
GET sensor_1001_last_reading
```

Podatkovna zbirka takoj vrne shranjeno vrednost brez filtriranja ali agregiranja. Učinkovitost tega mehanizma izhaja iz dejstva, da Redis uporablja shranjevanje v pomnilniku in indeksiranje na podlagi zgoščevalnih funkcij, kar omogoča pridobivanje v konstantnem času.

Sistemi ključ–vrednost se uporabljajo tudi za števce, denimo za spremljanje števila zaznav vrst v kratkem časovnem oknu. Na primer:

```
INCR site_001_detection_counter
```

Ta ukaz atomsko poveča števec, povezan s ključem. Atomska narava operacije zagotavlja pravilnost tudi takrat, ko števec hkrati posodablja več procesov, kar je bistveno v sistemih visokofrekvenčnih ekoloških dogodkov.

Shrambe ključ–vrednost sicer zagotavljajo izjemno hitrost in preprostost, vendar imajo omejene analitične zmožnosti. Ne morejo učinkovito podpirati poizvedb, kot je »poišči vse senzorje s temperaturo nad 30 °C«, ker so vrednosti za sistem neprosojne, če niso uvedene dodatne plasti indeksiranja. Zato je shrambe ključ–vrednost najbolje uporabljati kot dopolnilne sestavine v širših poliglotskih arhitekturah.

Dokumentno usmerjene podatkovne zbirke shranjujejo podatke v prilagodljivih, polstrukturiranih oblikah, kot so JSON, BSON ali XML. Vsak *dokument* zajema vse informacije o entiteti, vključno z ugnezdenimi objekti in polji. V nasprotju s sistemi ključ–vrednost dokumentne podatkovne zbirke razumejo notranjo strukturo shranjenih zapisov, zato omogočajo indeksiranje in poizvedovanje na podlagi ugnezdenih atributov.

Možni primeri uporabe vključujejo sisteme za upravljanje vsebin, kataloge izdelkov v e-trgovini, uporabniške profile in dnevnike dogodkov. Med dokumentno usmerjene podatkovne zbirke spadajo naslednji primeri. *MongoDB* je najbolj razširjena dokumentna podatkovna zbirka ter podpira prilagodljive sheme in bogate poizvedbe. *CouchDB* uporablja JSON za dokumente in HTTP za svoj API, pri čemer je poudarek na podvajanju in zasnovi offline-first. *ArangoDB* je večmodelna podatkovna zbirka, ki podpira dokumentne podatke, podatke ključ–vrednost in grafovne podatke. Značilnosti dokumentno usmerjenih podatkovnih zbirk vključujejo:

- podatkovni model brez sheme omogoča agilnost in hitro iteriranje;

- bogati jeziki za poizvedovanje in indeksiranje (ugnezdenih polj);
- omogočajo vodoravno razširjanje s particioniranjem (sharding) in nabori replik.

V okoljskih aplikacijah so dokumentne podatkovne zbirke zelo primerne za shranjevanje polstrukturiranih ekoloških opazovanj, pri katerih se lahko struktura zapisov med lokacijami razlikuje ali se sčasoma spreminja. Ena merilna postaja lahko na primer beleži temperaturo in vlažnost, druga pa tudi slanost in sevanje UV.

Oglejmo si naslednji primer MongoDB:

```
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
});
```

Ta dokument predstavlja posamezno okoljsko opazovanje. MongoDB ga shrani brez zahteve po vnaprej določeni shemi. Če se pojavi nov parameter senzorja, denimo zračni tlak, ga je mogoče preprosto 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
});
```

Spreminjanje sheme ni potrebno. Ta prilagodljivost je ključna v ekoloških sistemih, v katerih se lahko parametri spremljanja sčasoma spreminjajo zaradi spreminjajočih se raziskovalnih ciljev.

Dokumentne podatkovne zbirke omogočajo tudi indeksiranje določenih polj:

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

Ta sestavljeni indeks omogoča učinkovito pridobivanje nedavnih meritev za določeno lokacijo. Prvo polje (site_id) podatke logično razdeli glede na lokacijo, drugo polje (timestamp) pa omogoča kronološko razvrščanje. Ko se izvede poizvedba, kot je:

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

MongoDB uporabi indeks, da hitro vrne najnovejše zapise, ne da bi pregledal celotno zbirko.

Dokumentni model uravnoteža prilagodljivost z izrazno močjo poizvedb, zato je ena najširše sprejetih paradigem NoSQL v okoljskih sistemih.

Shrambe družin stolpcev organizirajo podatke v stolpce namesto v vrstice. Vsaka *družina stolpcev* združuje sorodne podatke za učinkovit dostop. Primeri uporabe vključujejo shranjevanje podatkov časovnih vrst, agregiranje dnevnikov in dogodkov ter obsežno analitiko in podatkovno skladiščenje. Primeri shramb družin stolpcev so naslednji. *Apache Cassandra* zagotavlja visoko razpoložljivost, linearno razširljivost in odpornost proti napakam. *HBase* je odprtokodna implementacija sistema Bigtable, zgrajena na porazdeljenem datotečnem sistemu HDFS platforme Hadoop. *ScyllaDB* je visokozmogljiva ponovna implementacija sistema Cassandra v jeziku C++. Najpomembnejše značilnosti shramb družin stolpcev so:

- odlična zmogljivost zapisovanja in razširljivost;
- nastavljive ravni doslednosti;
- optimiziranost za širokostolpčno shranjevanje;
- primernost za analitične delovne obremenitve.

Ta model je posebej učinkovit za okoljske podatke časovnih vrst, pri katerih se meritve zapisujejo zaporedno in se poizvedbe pogosto izvajajo po časovnem razponu.

Oglejmo si definicijo tabele v sistemu Cassandra:

```
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 je tukaj sestavljen iz particijskega ključa (site_id, day) in stolpca za razvrščanje timestamp. Particijski ključ določa, kako so podatki porazdeljeni med vozlišči gruče. Vse meritve za določeno lokacijo in dan so shranjene skupaj v istem vozlišču, kar zagotavlja učinkovite poizvedbe po razponu za dnevne povzetke.

Stolpec za razvrščanje določa fizični vrstni red vrstic znotraj particije. Ker so časovni žigi urejeni padajoče, je pridobivanje najnovejšega odčitka učinkovito:

```
SELECT *
```

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

Ta poizvedba ne pregleda celotnega podatkovnega niza. Cassandra namesto tega neposredno dostopa do ustrezne particije in na podlagi vrstnega reda razvrščanja pridobi najnovejše vnose.

Podatkovne zbirke z družinami stolpcev se odlikujejo v scenarijih sprejema podatkov z visoko prepustnostjo. Ko na tisoče senzorjev vsako sekundo pošilja odčitke, Cassandra razporedi zapisovanje med vozlišča s konsistentnim razprševanjem in tako prepreči ozka grla. Zato je posebej primerna za infrastrukture spremljanja okolja na nacionalni ravni.

Grafovne podatkovne zbirke uporabljajo grafovne strukture z vozlišči, povezavami in lastnostmi za predstavljanje in shranjevanje podatkov. Namenjene so raziskovanju razmerij in vzorcev med entitetami. Primeri njihove uporabe vključujejo družbena omrežja in priporočilne sisteme, odkrivanje prevar in analizo omrežij ter grafe znanja in semantične podatkovne modele.

Med primere grafovnih podatkovnih zbirk spadajo naslednje. *Neo4j* je najpogosteje uporabljena grafovna podatkovna zbirka in uporablja jezik za poizvedovanje Cypher. *Amazon Neptune* je upravljana grafovna storitev, ki podpira grafe z lastnostmi in grafe RDF. *OrientDB* združuje dokumentne in grafovne modele ter tako zagotavlja hibridno prilagodljivost. Značilnosti grafovnih podatkovnih zbirk vključujejo:

- optimizacijo za prehajanje po kompleksnih razmerjih;
- zmogljive jezike za poizvedovanje za iskanje poti in ujemanje vzorcev;
- primernost za povezane podatke, pri katerih so razmerja enako pomembna kot same entitete.

Grafovne podatkovne zbirke modelirajo podatke kot vozlišča, povezana z razmerji (robovi), pri čemer lahko tako vozlišča kot razmerja vsebujejo lastnosti. Ta predstavitev se naravno ujema z ekološkimi sistemi, kjer so razmerja med entitetami pogosto enako pomembna kot entitete same.

Vrste na primer naseljujejo lokacije, lokacije pripadajo regijam, regije pa so lahko del omrežij zavarovanih območij. Predstavitev teh interakcij v relacijskih tabelah zahteva več operacij združevanja, ki postajajo računsko zahtevnejše z večanjem globine razmerij.

V Neo4j je mogoče ta razmerja izrecno 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);
```

Vsak stavek MERGE zagotovi obstoj vozlišča ali razmerja, ne da bi ustvaril dvojnike. Izrecni razmerji OBSERVED_AT in LOCATED_IN omogočata intuitivne poizvedbe s prehajanjem.

Za izračun biotske raznovrstnosti po regijah je mogoče izvesti naslednjo poizvedbo:

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

Ta poizvedba prehaja po grafovni strukturi, namesto da bi izvajala relacijska združevanja. Ker grafovne podatkovne zbirke neposredno shranjujejo razmerja sosednosti, ostajajo operacije prehajanja učinkovite tudi v kompleksnih omrežjih.

Grafovni modeli so posebej uporabni za:

- modeliranje prehranjevalnih spletov,
- analizo migracijskih omrežij,
- ocenjevanje povezanosti habitatov,
- preučevanje interakcij v ekosistemih.

Njihova prednost je analitika, osredotočena na razmerja, ne pa obsežno agregiranje številskih podatkov.

2.3. NoSQL v primerjavi z relacijskimi podatkovnimi zbirkami

V nasprotju z relacijskimi podatkovnimi zbirkami, ki zahtevajo normalizacijo in stroge opredelitve shem, podatkovne zbirke NoSQL spodbujajo denormalizirano zasnovo, ki jo usmerja aplikacija. Modeliranje podatkov je močno odvisno od vzorcev dostopa aplikacije. Ključna načela vključujejo:

- *Zasnovo prilagodite poizvedbam, ne zagotavljanju celovitosti podatkov*: podatkovni model naj odraža, kako se podatki najpogosteje berejo in zapisujejo.
- *Sprejmite denormalizacijo*: podatke shranjujte podvojeno, da zmanjšate število kompleksnih združevanj in izboljšate zmogljivost.
- *Uporabljajte agregate*: sorodne podatke (npr. vse podrobnosti naročila) združite v en dokument ali družino stolpcev.
- *Vdelava ali sklicevanje*: izberite med vdelavo povezanih podatkov (hitrejše branje) in sklicevanjem nanje (boljša normalizacija).

Medtem ko SQL zagotavlja enoten poizvedovalni jezik, sistemi NoSQL uporabljajo različne mehanizme poizvedovanja:

1. Iskanje po ključih v shrambah ključ–vrednost.
2. Poizvedbe, podobne JSON, v dokumentnih podatkovnih zbirkah.
3. CQL (Cassandra Query Language) v shrambah družin stolpcev.
4. Jeziki za prehajanje grafov, kot sta Cypher in Gremlin, v grafovnih podatkovnih zbirkah.

Za optimizacijo zmogljivosti poizvedb se uporabljajo mehanizmi indeksiranja, kot so zgoščevalni indeksi, indeksi B-dreves in indeksi za iskanje po celotnem besedilu. Nekateri sistemi (npr. MongoDB) podpirajo sestavljene, geoprostorske in besedilne indekse, drugi (npr. Cassandra) pa za hitro pridobivanje podatkov uporabljajo particijske ključe in stolpce za gručenje. Sodobni sistemi pogosto uporabljajo vztrajnost poliglotskih podatkov, torej uporabo več tehnologij podatkovnih zbirk znotraj ene aplikacije. Vsaka komponenta uporablja najprimernejšo podatkovno shrambo: Redis za predpomnjenje in upravljanje sej, MongoDB za uporabniške profile in vsebine, Cassandra za visokoprepustne podatke o dogodkih ter Neo4j za priporočilno logiko. Integracija poteka prek API-jev, podatkovnih cevovodov ali čakalnih vrst sporočil. Ponudniki storitev v oblaku, kot so AWS in Azure, ponujajo upravljane rešitve NoSQL (npr. DynamoDB, Cosmos DB), ki podpirajo večmodelno shranjevanje in nemoteno razširjanje. Čeprav podatkovne zbirke NoSQL ponujajo izjemne prednosti, prinašajo tudi nove izzive, med katere spadajo:

- Zahtevnost upravljanja doslednosti v porazdeljenih sistemih.
- Kompromisi pri modeliranju podatkov, ki zahtevajo poglobljeno razumevanje vzorcev dostopa.
- Pomanjkanje standardizacije med poizvedovalnimi jeziki in vmesniki API.
- Upravljanje varnosti in transakcij, ki se razlikuje od pravil sistemov RDBMS.

Nastajajoči trendi si prizadevajo premostiti te vrzeli. Podatkovne zbirke NewSQL, kot sta CockroachDB in Google Spanner, združujejo razširljivost sistemov NoSQL s transakcijami ACID. Večmodelne podatkovne zbirke, kot sta ArangoDB in OrientDB, pa razvijalcem omogočajo delo z dokumenti, grafi in pari ključ–vrednost v enem samem pogonu. Prihodnost nerelacijskega upravljanja podatkov je v hibridnih arhitekturah – sistemih, ki se lahko dinamično prilagajajo spreminjajočim se delovnim obremenitvam ter vključujejo optimizacijo poizvedb z umetno inteligenco, samodejno horizontalno particioniranje in interoperabilnost med modeli. NoSQL in nerelacijske podatkovne zbirke so na novo opredelile način shranjevanja in upravljanja podatkov v porazdeljenih, visoko zmogljivih okoljih. Omejitve tradicionalnih relacijskih sistemov odpravljajo s prilagodljivostjo, razširljivostjo in sposobnostjo prilagajanja različnim oblikam podatkov in

primerom uporabe. Tehnologije NoSQL so temelj današnjih najinovativnejših digitalnih ekosistemov, od predpomnilnikov ključ–vrednost in dokumentnih repozitorijev do stolpčne analitike in sklepanja na podlagi grafov. Razumevanje njihovih načel, kompromisov in praktičnih uporab je bistveno za sodobne programske inženirje in podatkovne arhitekte, ki gradijo sisteme za podatkovno intenziven svet.

2.4. Podatkovne zbirke NoSQL v okoljskem kontekstu

Tradicionalno relacijsko načrtovanje se začne z opredelitvijo tabel, razmerij in omejitev, še preden je implementirana aplikacijska logika. Sistemi NoSQL pa uporabljajo filozofijo modeliranja, ki izhaja iz aplikacije. Namesto vprašanja, kako naj bodo podatki normalizirani, arhitekti najprej analizirajo, kako se bo do podatkov dostopalo, kako se bo po njih poizvedovalo in kako se bo sistem razširjal. Ta premik je posebej pomemben v okoljskih sistemih, kjer se poizvedbe pogosto osredotočajo na nedavne meritve senzorjev, določena prostorska območja ali agregirane kazalnike ekosistema. Načrtovanje za takšne vzorce dostopa zahteva prilagodljivost strukture in porazdelitve.

Oglejmo si na primer preprosto relacijsko shemo za spremljanje okolja:

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

Čeprav se ta shema na prvi pogled zdi zadostna, nastanejo zapleti, ko uvedemo nove okoljske spremenljivke. Dodajanje novih atributov zahteva spremembe sheme:

```
ALTER TABLE sensor_readings
```

```
ADD air_pressure FLOAT;
```

V hitro razvijajočih se infrastrukturah za spremljanje, kjer se senzorji pogosto nadgrajujejo ali razširjajo, lahko ponavljajoče se spremembe sheme povzročajo motnje in visoke operativne stroške.

Nasprotno pa dokumentno usmerjena podatkovna zbirka, kot je MongoDB, omogoča shranjevanje istega zapisa senzorja brez vnaprej določenih omejitev 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
});
```

Tukaj je struktura prilagodljiva. Če se pojavi nov parameter, ga je mogoče preprosto vključiti v prihodnje dokumente, ne da bi spreminjali obstoječe zapise. Ta prilagodljivost ponazarja eno temeljnih prednosti sistemov NoSQL v dinamičnih okoljskih sistemih.

Poleg prilagodljivosti sheme je še ena ključna razlika porazdeljena razširljivost. Platforme za spremljanje okolja pogosto zbirajo podatke iz več sto ali tisoč geografsko razpršenih senzorjev. Centralizirana podatkovna zbirka pri takšni obremenitvi hitro postane ozko grlo. Porazdeljeni sistemi NoSQL to rešujejo z razdeljevanjem podatkov med več vozlišč. V sistemu Cassandra se na primer podatki porazdeljujejo s particijskimi ključi:

```
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);
```

Sestavljeni particijski ključ (site_id, day) zagotavlja, da so dnevne meritve za posamezno lokacijo združene, hkrati pa enakomerno porazdeljene med vozlišči gruče. Ko se omrežju za spremljanje dodajajo nove lokacije, je mogoče v gručo vključiti nova vozlišča brez prestrukturiranja sheme, s čimer je omogočena vodoravna razširljivost.

Porazdeljeni sistemi pa uvajajo kompromise. Izrek CAP kaže, da ob razdelitvah omrežja sistem ne more hkrati zagotavljati stroge doslednosti in popolne razpoložljivosti. Številni okoljski sistemi dajejo prednost razpoložljivosti, zlasti v primerih spremljanja v izrednih razmerah. Nizi replik MongoDB so na primer konfigurirani tako, da zagotavljajo redundanco:

```
                                                rs.initiate({
                                                   _id: "replicaSet1",
                                                   members: [
                                                      { _id: 0, host:
"node1:27017" },
                                                      { _id: 1, host:
"node2:27017" },
                                                      { _id: 2, host:
"node3:27017" }
                                                   ]
                                                });
```

V tej konfiguraciji je ob odpovedi primarnega vozlišča samodejno izbrano sekundarno vozlišče, kar zagotavlja neprekinjeno razpoložljivost sistema. Med preklopom ob odpovedi se lahko pojavi

začasna nedoslednost podatkov, vendar sistem ostane operativen – kar je bistvena zahteva za ekološko spremljanje v realnem času.

Različni modeli NoSQL zagotavljajo strukturne abstrakcije, prilagojene različnim vrstam okoljskih podatkov. Dokumentne podatkovne zbirke obravnavajo polstrukturirana ekološka opazovanja, shrambe ključ–vrednost omogočajo izjemno hitro predpomnjenje za nadzorne plošče v realnem času, podatkovne zbirke z družinami stolpcev pa podpirajo visokoprepusten zajem podatkov časovnih vrst. Grafovne podatkovne zbirke so posebej dragocene za modeliranje ekoloških razmerij. V sistemu Neo4j je na primer mogoče vrste in habitate izrecno povezati:

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

Ta predstavitev omogoča zapletene poizvedbe za prehajanje grafa, kot je prepoznavanje vrst, opaženih v medsebojno povezanih habitatih:

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

Ko se globina razmerij poveča, je takšne operacije na podlagi grafov težko učinkovito izvesti v tradicionalnih relacijskih sistemih.

Sodobne okoljske platforme se v praksi redko opirajo na eno samo paradigmo podatkovnih zbirk. Namesto tega uporabljajo poliglotsko persistenco, pri kateri v enotni arhitekturi združujejo več specializiranih sistemov. Zajem podatkov v realnem času lahko na primer obravnava Kafka, odčitki senzorjev se shranjujejo v sistemu Cassandra, metapodatki se vzdržujejo v sistemu MongoDB, analiza razmerij pa se izvaja v sistemu Neo4j. Integracija med komponentami poteka prek vmesnikov API ali pretočnih cevovodov, ne pa z monolitnimi relacijskimi združevanji.

Ta arhitekturna strategija omogoča, da vsak podatkovni model deluje skladno s svojimi prednostmi, hkrati pa se ohranja celovita usklajenost sistema. S porazdelitvijo odgovornosti med specializirane mehanizme sistem doseže večjo razširljivost, boljšo odpornost proti napakam in večjo prilagodljivost spreminjajočim se okoljskim zahtevam.

Temelji sistemov NoSQL in porazdeljenih podatkovnih arhitektur navsezadnje odražajo širši razvoj upravljanja okoljskih podatkov. Namesto uveljavljanja centraliziranih in togih shem sodobni sistemi uporabljajo porazdeljene, razširljive in prilagodljive zasnove, ki ustrezajo dejanskim razmeram visokofrekvenčnih senzorskih omrežij in heterogenih ekoloških podatkovnih nizov. Razumevanje teh načel je bistveno za načrtovanje odpornih platform, ki lahko podpirajo spremljanje v realnem času, dolgoročne analize in trajnostno okoljsko odločanje.

3. Študija primera: platforma za spremljanje biotske raznovrstnosti in okolja v realnem času

3.1 Konceptualni okvir in cilji

Ta študija primera prikazuje, kako lahko porazdeljena arhitektura NoSQL podpira ne le razširljivo shranjevanje okoljskih podatkov, temveč tudi strukturirano ekološko razlago. **Cilj** je zasnovati in implementirati platformo za spremljanje biotske raznovrstnosti in okolja v realnem času, ki lahko heterogena okoljska opazovanja pretvori v prepoznavne strukture ekoloških režimov.

Sistem združuje visokofrekvenčne podatke okoljskih senzorjev, meteorološke meritve časovnih vrst in zapise o pojavnosti biotske raznovrstnosti. Ti podatkovni viri se razlikujejo po strukturi, časovni ločljivosti in pomenu. Osrednji izziv je zato njihovo usklajevanje v enotnem analitičnem delovnem toku brez zmanjšanja razširljivosti, prilagodljivosti ali razložljivosti.

Spoznajmo logiko razvoja analitičnega cevovoda, ki poteka od porazdeljenega zajema in shranjevanja podatkov do oblikovanja značilk in nenadzorovanega prepoznavanja režimov. Končni cilj je na podlagi združenih okoljskih kazalnikov in kazalnikov biotske raznovrstnosti zaznati strukturna ekološka stanja (stabilna, prehodna in kritična).

3.2 Podatkovne domene in strukturne značilnosti

Platforma za spremljanje združuje tri glavne podatkovne domene.

Prvo domeno sestavljajo okoljske meritve na podlagi interneta stvari (IoT). Vlažnost tal, temperatura površja in padavine se na več mokriščnih lokacijah beležijo z minutno ločljivostjo. Te meritve predstavljajo obsežne podatke časovnih vrst s strogim časovnim zaporedjem.

Druga domena vključuje meteorološke podatke, ki se vsako uro pridobivajo iz zunanjega vmesnika API. Ti podatki opisujejo atmosferske razmere, ki vplivajo na vzorce hidrološkega stresa.

Tretjo domeno sestavljajo zapisi o pojavnosti biotske raznovrstnosti, uvoženi iz repozitorijev opazovanj vrst. Ti zapisi so polstrukturirani dokumenti, ki vsebujejo identifikatorje vrst, geografske koordinate, časovne žige in kontekstualne metapodatke.

Vsaka domena ima različne strukturne lastnosti in pogostost posodabljanja, kot prikazuje tabela 1.

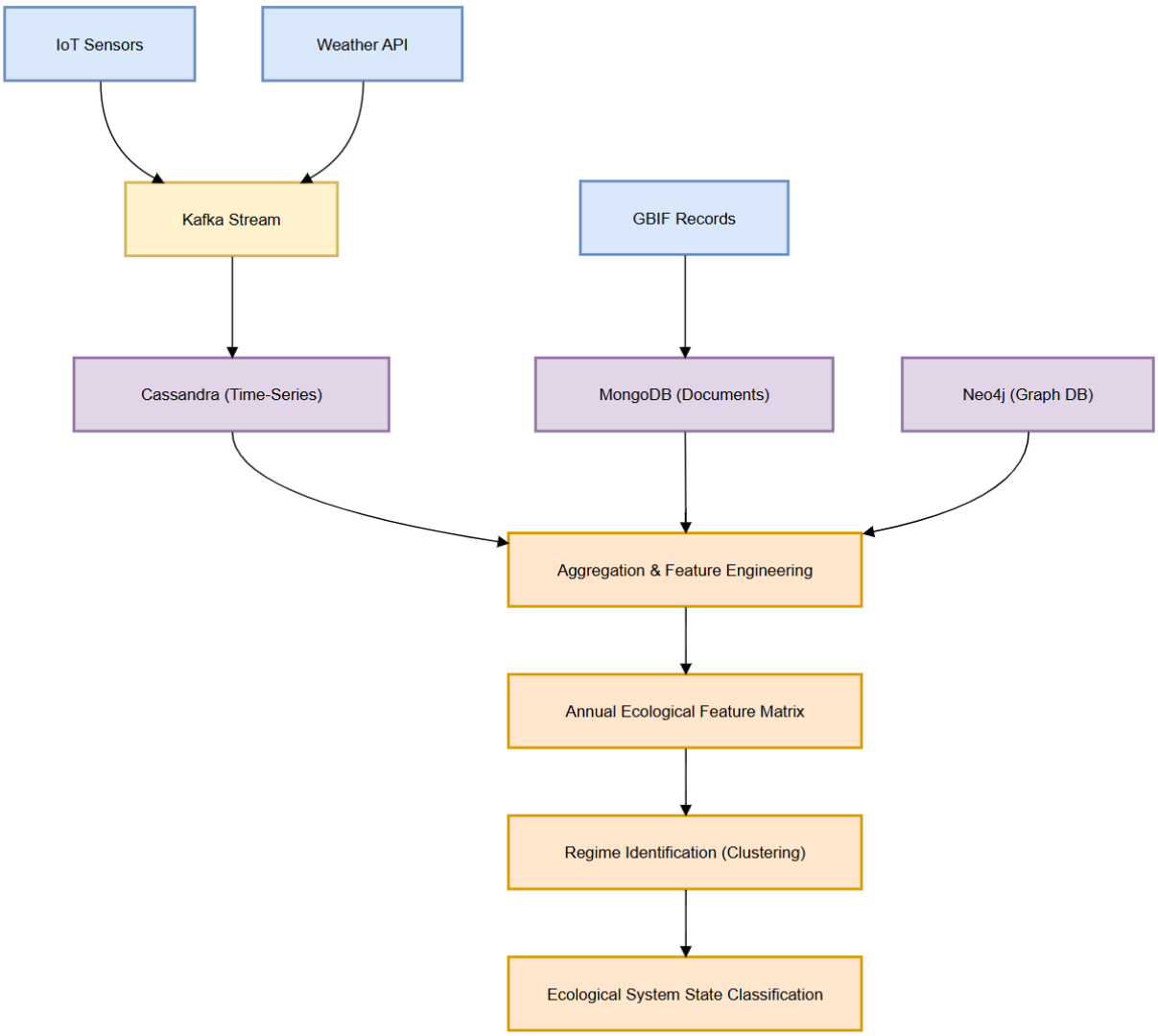
Tabela 1. Strukturne značilnosti in ekološke vloge združenih podatkovnih domen

Podatkovna domena	**Podatkovna struktura**	**Časovna ločljivost**	**Ekološka vloga**
Okoljski senzorji IoT	Časovne vrste	Minutna	Lokalni hidrološki in toplotni stres
Meteorološki API	Časovne vrste	Urna	Podnebno siljenje
Zapisi o biotski raznovrstnosti	Polstrukturirani dokumenti	Na podlagi dogodkov	Biološki odziv
Razmerja med vrstami in lokacijami	Grafovna struktura	Dogodkovno vodena	Ekološka povezanost

Ta heterogenost zahteva strategijo poliglotske persistence, pri kateri se glede na strukturne in analitične zahteve uporabljajo različni modeli NoSQL.

3.3 Porazdeljena arhitektura in tok podatkov

Oglejmo si porazdeljeno arhitekturo in celoten tok podatkov, ki sta konceptualno prikazana na sliki 1.



Slika 1. Arhitektura platforme in tok podatkov

Arhitektura ločuje zajem podatkov, specializacijo shranjevanja, integracijo in analitično modeliranje v različne, vendar medsebojno povezane plasti.

Tokovi podatkov IoT in meteoroloških podatkov se najprej shranijo v medpomnilnik prek *pretočne plasti, ki temelji na sistemu Kafka*. S tem se ustvarjalci podatkov ločijo od sistemov za shranjevanje, odpornost pa se poveča z zagotavljanjem začasne hrambe podatkov med prekinitvami sistema.

Okoljski podatki časovnih vrst se shranjujejo v sistemu *Cassandra*. Njegova particijska arhitektura podpira vodoravno razširljivost in veliko prepustnost zapisovanja, zato je primerna za neprekinjen zajem meritev z minutno ločljivostjo.

Zapisi o pojavnosti biotske raznovrstnosti se shranjujejo v sistemu *MongoDB*. Dokumentni model omogoča prilagodljivo predstavitev spreminjajočih se atributov vrst in podpira geoprostorsko indeksiranje za prostorske poizvedbe.

Ekološka razmerja med vrstami, lokacijami in regijami se modelirajo v sistemu *Neo4j*. Grafovna predstavitev omogoča učinkovite poizvedbe za prehajanje grafa, ki zajamejo vzorce ekološke povezanosti.

Ta večplastna arhitektura zagotavlja, da vsak sistem za shranjevanje izvaja naloge, usklajene z njegovimi strukturnimi prednostmi, hkrati pa prispeva k enotnemu analitičnemu rezultatu.

3.4 Načrtovanje shranjevanja in agregiranje kazalnikov

Kot je pojasnjeno zgoraj, se okoljske meritve časovnih vrst shranjujejo v sistemu Cassandra. To je mogoče izvesti z uporabo sestavljenega particijskega ključa:

```
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) porazdeli podatke med vozlišči, hkrati pa lokalizira meritve posameznega dne. Stolpec za gručenje timestamp zagotavlja kronološko razvrščanje znotraj vsake particije.

Za izpeljavo letnih okoljskih kazalnikov agregacijske poizvedbe strnejo podatke z minutno ločljivostjo v strukturirane spremenljivke stresa:

```
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 kazalniki količinsko opredeljujejo toplotni stres, primanjkljaj padavin in pogostost hidrološkega stresa.

Zapisi o pojavnosti biotske raznovrstnosti se shranjujejo v sistemu MongoDB. Podatki o pojavnosti vrst se shranjujejo kot prilagodljivi dokumenti JSON:

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

Vrstno bogastvo za posamezno lokacijo in leto se izračuna z agregacijskim cevovodom:

```
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" }
    }
  }
])
```

Ta postopek ustvari kazalnike odziva biotske raznovrstnosti, usklajene s spremenljivkami okoljskega stresa.

3.5 Izdelava letne matrike ekoloških značilk

Agregirani okoljski kazalniki in kazalniki biotske raznovrstnosti se združijo v enotno analitično predstavitev: letno matriko ekoloških značilk (tabela 2). V tej matriki vsaka vrstica ustreza opazovanju za posamezno lokacijo in leto, vsak stolpec pa predstavlja izpeljani ekološki atribut.

Tabela 2. Letna matrika ekoloških značilk

Lokacija	Leto	Povprečna temperatura	Sušni dnevi	Dogodki suše	Vrstno bogastvo
Wetland A	2018	14.2	210	2	58

Wetland A	2019	15.6	242	5	47
Wetland A	2020	17.1	275	9	32

Ta matrika združuje okoljski stres in odziv biotske raznovrstnosti v eni sami opazovalni enoti. Pred razvrščanjem v gruče se značilke standardizirajo, da se zagotovi njihov uravnotežen prispevek:

```
from pyspark.ml.feature import StandardScaler

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

Ta standardizacija zagotavlja, da zaradi razlik v merilnih lestvicah nobena posamezna spremenljivka ne prevlada v postopku razvrščanja v gruče.

3.6 Prepoznavanje in analiza struktur ekoloških režimov

Za prepoznavanje strukturnih ekoloških režimov se na standardizirani matriki značilk izvede nenadzorovano razvrščanje v gruče.

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

Centroidi gruč razkrivajo tri ekološke režime, prikazane v tabeli 3.

Tabela 3. Ekološki režimi

Gruča	Povprečna temperatura	Dogodki suše	Vrstno bogastvo	Razlaga režima
1	Zmerna	Redki	Visoko	Stabilen
2	Povišana	Občasni	Zmerno	Prehoden
3	Visoka	Pogosti	Nizko	Kritični stres

Časovna porazdelitev gruč kaže premike med režimi, kot je prikazano v tabeli 4.

Tabela 4. Gruče po letih

Leto	Gruča	Leto	Gruča	Leto	Gruča
2015	1	2018	2	2021	3
2016	1	2019	3	2022	3
2017	2	2020	3	2023	3

Prehodna matrika potrjuje odvisnost od poti, kot je prikazano v tabeli 5.

Tabela 5. Prehodna matrika

Iz \ V	Gruča 1	Gruča 2	Gruča 3
Gruča 1	1	1	0
Gruča 2	0	1	1
Gruča 3	0	0	4

Ko ekosistem preide v kritični režim, se poveča obstojnost tega režima, kar kaže na zmanjšano odpornost ekosistema. Te informacije bi lahko bile ključne pri odločanju o upravljanju ekosistemov.

Študija primera tako kaže, da sistemi NoSQL niso zgolj razširljive rešitve za shranjevanje podatkov. V kombinaciji s strukturiranim agregiranjem in oblikovanjem značilk omogočajo večrazsežno ekološko modeliranje. Povezovanje kazalnikov okoljskega stresa v časovnih vrstah in meril odziva biotske raznovrstnosti razkriva strukturne ekološke režime, ki odražajo trajne preobrazbe okolja.

S specializacijo porazdeljenega shranjevanja, integriranim oblikovanjem značilk in prepoznavanjem režimov platforma za spremljanje pretvarja neobdelane okoljske podatke v razumljiva stanja sistema. Ta stanja zagotavljajo uporabne vpoglede za ekološko upravljanje, ocenjevanje tveganj in zaznavanje zgodnjih opozorilnih znakov.

Praktične vaje

Naloga 1

1. Sestavite heterogen nabor okoljskih in ekoloških podatkov z uporabo virov, navedenih v spodnji tabeli.

Podatkovni niz	Opis	Povezava
GBIF (Global Biodiversity Information Facility)	Obsežno skladišče podatkov o pojavljanju vrst in opazovanjih biotske raznovrstnosti	https://www.gbif.org/
European Environment Agency – DataHub	Okoljski in ekološki podatkovni nizi iz vse Evrope	https://www.eea.europa.eu/en/datahub
UKCEH (UK Centre for Ecology & Hydrology)	Podatkovni nizi za spremljanje okolja; metapodatki o časovnih vrstah in prostorski metapodatki	https://www.ceh.ac.uk/data
NEON (National Ecological Observatory Network)	Ekološki podatkovni viri za odprto znanost	https://libguides.cmich.edu/lifesciencedata/ecological

2. Analizirajte značilnosti sestavljenega podatkovnega niza.

3. Priporočite arhitekturo podatkovne shrambe NoSQL za delo s tem podatkovnim nizom.

Rezultat: poročilo o rezultatih.

Merila ocenjevanja: popolnost rezultatov in jasnost razlage.

Naloga 2

Predstavljajte si, da sodelujete z okoljsko agencijo, ki spremlja pojavljanje vrst, habitatne parametre (temperaturo, vlažnost zraka in vlažnost tal) ter dogodkovna opazovanja (npr. sprožitve fotopasti in alarme senzorjev). Predlagajte tehnološki sklad NoSQL za zasnovo programskega sistema za spremljanje, ki omogoča zbiranje podatkovnih tokov iz senzorjev in fotopasti na številnih lokacijah, prilagodljivo poizvedovanje ter analizo ekoloških parametrov.

Rezultat: poročilo o rezultatih.

Merila ocenjevanja: popolnost rezultatov in jasnost razlage.

Preverjanje znanja

1. Rešite križanko.

Izrek CAP pravi, da v porazdeljenem podatkovnem sistemu ni mogoče hkrati zagotoviti vseh treh naslednjih lastnosti:

C																		
A																		
P									—									

2. Odgovorite na vprašanja v kvizu.

1. Katera od naslednjih trditev najbolje opisuje *glavni razlog* za uporabo podatkovnih zbirk NoSQL?

A. Strogo uveljavljanje omejitev relacijske celovitosti

B. Učinkovita obdelava obsežnih nestrukturiranih ali polstrukturiranih podatkov

C. Poenostavitev zasnove relacijske sheme

D. Odprava potrebe po sistemih za upravljanje podatkovnih zbirk

Pravilen odgovor: B

2. Kaj od naslednjega *ni* vrsta podatkovne zbirke NoSQL?

A. Shramba ključ–vrednost

B. Dokumentna podatkovna zbirka

C. Shramba družin stolpcev

D. Shramba relacijskih tabel

Pravilen odgovor: D

3. V *dokumentno usmerjeni podatkovni zbirki NoSQL* so podatki običajno shranjeni v:

A. datotekah CSV

B. dokumentih, podobnih JSON-u

C. zapisih fiksne dolžine

D. tabelah s tujimi ključi

Pravilen odgovor: B

4. Katera od naslednjih podatkovnih zbirk NoSQL je primer *shrambe ključ–vrednost*?

A. MongoDB

B. Cassandra

C. Redis

D. Neo4j

Pravilen odgovor: C

5. Kateri model NoSQL je najprimernejši za predstavitev *omrežnih podatkov ali podatkov, ki temeljijo na razmerjih*, kot so grafi družbenih omrežij ali ekološke interakcije med vrstami?

A. Shramba družin stolpcev

B. Shramba ključ–vrednost

C. Grafovna podatkovna zbirka

D. Dokumentna podatkovna zbirka

Pravilen odgovor: C

6. Katera je glavna prednost *vodoravnega razširjanja* v sistemih NoSQL?

A. Z normalizacijo vseh tabel zmanjšuje podvajanje podatkov.

B. Omogoča povečanje zmogljivosti sistema z dodajanjem vozlišč namesto z nadgrajevanjem strojne opreme.

C. Zagotavlja popolno skladnost z načeli ACID v porazdeljenih sistemih.

D. Nestrukturirane podatke samodejno pretvori v relacijsko obliko.

Pravilen odgovor: B

7. Katera komponenta v ekosistemu NoSQL je odgovorna za *zajemanje neprekinjenih podatkovnih tokov* iz senzorjev ali naprav IoT?

A. Mehanizem za indeksiranje

B. Procesor poizvedb

C. Sloj za zajem podatkov / pretočni sloj

D. Nadzorna plošča za poročanje

Pravilen odgovor: C

8. Pri aplikacijah za ekološko spremljanje je *podatkovna zbirka časovnih vrst* posebej uporabna za:

A. upravljanje podatkov s kompleksnimi odnosi

B. shranjevanje zaporednih odčitkov senzorjev in okoljskih podatkov s časovnimi žigi

C. shranjevanje nestrukturiranih besedilnih podatkov

D. izvajanje iskanja po celotnem besedilu v dokumentih

Pravilen odgovor: B

9. Katera od naslednjih tehnologij se pogosto uporablja za *pretočne podatkovne cevovode v realnem času* v arhitekturah, ki temeljijo na NoSQL?

A. Apache Kafka

B. Microsoft Access

C. Oracle RDBMS

D. SQLite

Pravilen odgovor: A

10. Katera od naslednjih trditev najbolje opisuje *dokumentno podatkovno zbirko*, kot je MongoDB?

A. Zahteva vnaprej določene sheme in združevanje tabel.

B. Podatke shranjuje in pridobiva v obliki prilagodljivih ugnezdenih dokumentov.

C. Optimizirana je za kompleksne relacijske poizvedbe z uporabo jezika SQL.

D. Ne more obdelovati hierarhičnih ali ugnezdenih podatkovnih struktur.

Pravilen odgovor: B

Merila ocenjevanja: število točk v kvizu (vsako vprašanje: 1 točka; skupaj: 10 točk).

Viri / dodatno branje

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.