MySql Clusters

Cândea Sebastian-Constantin

Rezumat

Această lucrare analizează conceptele fundamentale ale bazelor de date distribuite, punând accent pe soluția MySQL Cluster. Prin compararea arhitecturilor centralizate și distribuite, se evidențiază avantajele de scalabilitate, disponibilitate și toleranță la defecțiuni oferite de sistemele moderne. Lucrarea explorează mecanismele interne de replicare sincronizată, arhitectura multi-nod a MySQL Cluster și procedurile automatizate de gestionare a defecțiunilor. De asemenea, sunt prezentate aspecte practice legate de configurarea și testarea unui cluster, oferind o perspectivă aplicată complementară fundamentelor teoretice. Concluziile subliniază importanța acestei tehnologii în contextul aplicațiilor critice pentru mediul de afaceri.

1 Introducere

Bazele de date reprezintă fundația sistemelor informatice contemporane, evoluând de la modele centralizate simple către arhitecturi distribuite complexe. În contextul actual al volumelor mari de date, cerințele privind cantitatea, viteza de procesare și disponibilitatea informațiilor au impus o regândire a modelelor tradiționale. Sistemele centralizate, deși ușor de gestionat în medii cu trafic redus, devin nepotrivite în scenarii unde o singură defecțiune poate compromite întreaga infrastructură. Conform lucrării „Principii ale Sistemelor de Baze de Date Distribuite" [1], scalabilitatea orizontală și eliminarea punctelor unice de eșec sunt esențiale pentru aplicațiile critice. MySQL Cluster, o soluție open-source dezvoltată de Oracle, răspunde acestor cerințe prin integrarea unui motor de stocare distribuit (NDB) și a mecanismelor avansate de replicare. Această lucrare își propune să ofere o înțelegere cuprinzătoare a acestei tehnologii, evidențiind diferențele dintre abordările tradiționale și cele distribuite și detaliind aspectele practice de implementare.

2 Baze de Date Centralizate versus Distribuite: O Analiză Comparativă

Bazele de date centralizate funcționează pe un singur server care gestionează toate operațiunile de stocare și procesare. Acest model, predominant în primele etape ale informatizării, oferă simplitate în administrare și consistență strictă datorită centralizării tranzacțiilor. Cu toate acestea, limitările sale devin evidente în mediile moderne: scalabilitatea verticală (adăugarea de resurse pe același server) este costisitoare și limitată fizic, iar dependența de un singur nod introduce riscuri majore de indisponibilitate. Un studiu realizat de Gartner în 2022 a arătat că peste 60% din perioadele de nefuncționare ale sistemelor centralizate sunt cauzate de defecțiuni hardware sau întreruperi de rețea.

În contrast, bazele de date distribuite împart datele pe mai multe noduri interconectate, oferind redundanță și capacitate de procesare paralelă. MySQL Cluster, de exemplu, permite distribuirea fragmentelor de date („shards") între noduri, fiecare fragment fiind replicat pe cel puțin două noduri pentru asigurarea toleranței la defecțiuni. Conform documentației oficiale [3], acest model asigură disponibilitate neîntreruptă chiar și în cazul în care un întreg centru de date devine nefuncțional, prin mecanisme de replicare geografică. Scalabilitatea orizontală permite adăugarea de noduri în timp real pentru a face față creșterilor neașteptate ale traficului, un avantaj esențial pentru aplicațiile web cu milioane de utilizatori simultani.

3 Concepte Teoretice în Sistemele Distribuite

3.1 Modelul CAP

Modelul CAP, formulat de Eric Brewer, afirmă că într-un sistem distribuit este imposibil să se garanteze simultan toate cele trei proprietăți: consistență (**C**onsistency), disponibilitate (**A**vailability) și toleranță la partiționare (**P**artition Tolerance). Sistemele distribuite trebuie să aleagă două dintre aceste trei proprietăți. MySQL Cluster prioritizează consistența și disponibilitatea, tolerând partiționări prin replicare sincronă și mecanisme automate de failover.

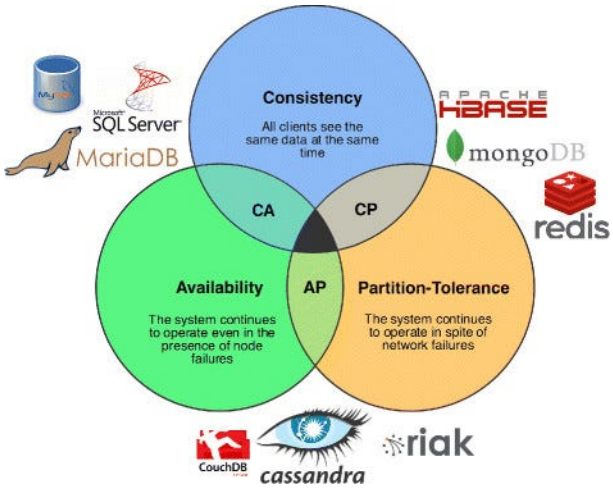

Figura 1: Modelul CAP. **Sursa**:[2]

3.2 Fragmentarea Datelor (Sharding)

Fragmentarea este procesul de împărțire a datelor în bucăți mai mici, numite fragmente sau *shards*, care sunt distribuite între noduri diferite. Fragmentarea poate fi:

- **Orizontală** – fiecare fragment conține un subset de rânduri dintr-o tabelă.
- **Verticală** – fiecare fragment conține un subset de coloane dintr-o tabelă.

În MySQL Cluster, datele sunt fragmentate orizontal și replicate automat pentru a asigura redundanța.

3.3 Protocolul Two-Phase Commit (2PC)

Protocolul *Two-Phase Commit* (2PC) este un mecanism utilizat pentru a asigura consistența atomică a tranzacțiilor distribuite în sistemele de baze de date. Este folosit atunci când o tranzacție implică mai multe noduri (participanți) care trebuie să fie de acord asupra rezultatului final: fie toate modificările sunt efectuate, fie niciuna nu este aplicată.

Fazele protocolului 2PC

Protocolul se desfășoară în două faze principale:

1. **Faza de pregătire (Prepare Phase):**
   - Coordonatorul (Transaction Manager) trimite un mesaj `PREPARE` tuturor participanților.
   - Fiecare participant verifică dacă poate efectua tranzacția și răspunde cu:
     - `VOTE_COMMIT` dacă totul este în regulă.
     - `VOTE_ABORT` dacă întâmpină o eroare.
2. **Faza de angajare (Commit Phase):**
   - Dacă toți participanții trimit `VOTE_COMMIT`, coordonatorul trimite mesajul `GLOBAL_COMMIT`.
   - Dacă cel puțin un participant trimite `VOTE_ABORT`, coordonatorul trimite mesajul `GLOBAL_ABORT`.
   - Participanții aplică decizia și confirmă încheierea procesului.

Reprezentare logică a 2PC

Coordonator	Participanți
Trimite `PREPARE`	
	Răspund cu `VOTE_COMMIT` sau `VOTE_ABORT`
Toate răspunsurile sunt `VOTE_COMMIT`?	
DA ⇒ Trimite `GLOBAL_COMMIT` NU ⇒ Trimite `GLOBAL_ABORT`	
	Execută commit sau rollback în funcție de decizie

Avantaje și Limitări

- **Avantaje:** Asigură consistența între mai multe baze de date.
- **Limitări:** Blocant în caz de eșec al coordonatorului, deoarece participanții rămân în stare de așteptare.

3.4 Modele de Consistență

Există mai multe modele de consistență în sistemele distribuite:

- **Consistență puternică** – toate nodurile văd aceleași date în orice moment.
- **Consistență eventuală** – nodurile se vor sincroniza în timp, dar pot returna valori diferite temporar.
- **Consistență cauzală** – păstrează ordinea logică a operațiunilor dependente.

MySQL Cluster oferă consistență puternică prin replicare sincronă, ceea ce asigură coerența datelor fără întârziere.

3.5 Transacții ACID în Sisteme Distribuite

Tranzacțiile distribuite trebuie să respecte proprietățile ACID (Atomicitate, Consistență, Izolare, Durabilitate). Pentru a implementa aceste cerințe, se folosesc protocoale precum *Two-Phase Commit*, care asigură că toate nodurile participante fie finalizează tranzacția, fie o anulează.

4 Arhitectura MySQL Cluster și Mecanisme de Replicare

MySQL Cluster se bazează pe motorul NDB (Network Database), conceput special pentru medii distribuite. Arhitectura sa cuprinde trei tipuri de noduri:

- **Noduri de management (mgmd):** Administrează metadatele clusterului și coordonează operațiunile de reconfigurare. Documentația oficială precizează că, deși aceste noduri sunt importante pentru administrare, clusterul poate funcționa temporar și fără ele [4].
- **Noduri de date (ndbd/ndbmtd):** Stochează fragmentele de date și replicile acestora. Fiecare operație de scriere este validată doar după ce cel puțin două noduri confirmă primirea, asigurând astfel consistența sincronă.
- **Noduri SQL (mysqld):** Acționează ca interfață pentru aplicații, transformând interogările SQL în operații pe nodurile de date.

Pentru o funcționare minimală, este necesar un nod de management, două noduri de date (pentru replicare) și cel puțin un nod SQL. Această configurație elimină punctele unice de eșec, deoarece fiecare componentă critică este redundantă.

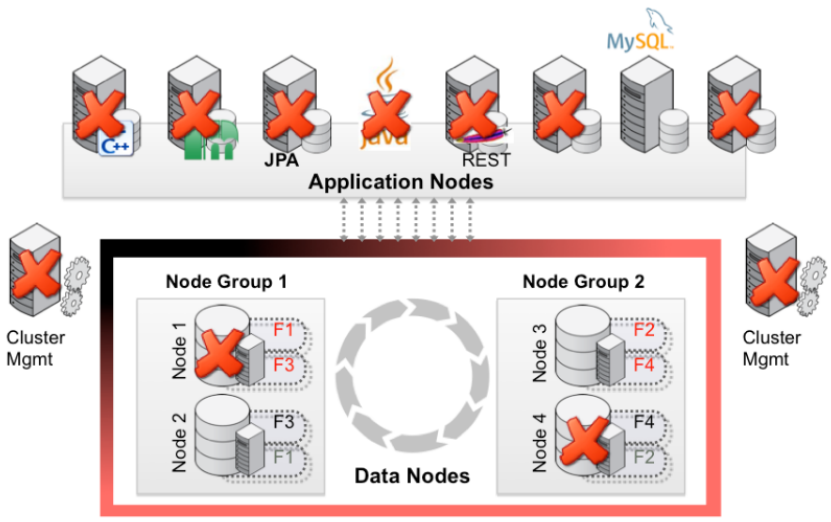

Figura 2: MySQL Cluster oferă rezistență extremă la defecțiuni, fără un singur punct de vulnerabilitate. **Sursa**: [3]

4.1 Replicarea Sincronizată versus Replicarea Asincronă

În replicarea clasică master-slave, actualizările sunt transmise asincron către servere slave prin intermediul jurnalelor binare (binlog). Acest model, deși simplu, introduce latență și risc de pierdere a datelor în caz de defectare a serverului master. MySQL Cluster schimbă acest paradigmă prin replicarea sincronizată: fiecare tranzacție este confirmată de nodurile de date înainte de a fi considerată finalizată. Conform documentației oficiale [5], acest mecanism reduce semnificativ riscul de inconsistență, deoarece toate replicile sunt actualizate în mod atomic.

Comutarea automată în caz de eșec (failover) reprezintă un alt avantaj esențial. În sistemele tradiționale, promovarea unui server slave la statutul de master necesită intervenție manuală și întreruperea temporară a serviciilor. În schimb, MySQL Cluster redirecționează automat traficul către nodurile funcționale în mod transparent, asigurând continuitatea serviciilor. De exemplu, dacă un nod de date eșuează, nodurile SQL detectează automat modificarea și redirecționează interogările către replica intactă.

5 Implementare și Testare a unui Cluster Tolerant la Defecțiuni

5.1 Configurarea Inițială

Primul pas constă în instalarea și configurarea nodurilor componente. Pentru implementarea MySQL Cluster pe sistemul de operare Windows, am descărcat arhiva oficială de pe site-ul MySQL și am dezarhivat-o în directorul C:\mysql-cluster. Structura de directoare creată pentru organizarea eficientă a clusterului cuprinde:

- **C:\mysql-cluster\config** - directorul pentru fișierele de configurare
- **C:\mysql-cluster\data** - directorul pentru stocarea datelor, cu subdirectoare separate pentru fiecare nod
- **C:\mysql-cluster\start** - directorul pentru scripturile de pornire

Fișierul de configurare al clusterului (config.ini) definește topologia completă a sistemului distribuit:

```
[NDBD DEFAULT]
NoOfReplicas=2

[ndb_mgmd]
hostname=mgmd
datadir=C:/mysql-cluster/data/mgmd
nodeid=1

[ndbd]
hostname=ndb1
datadir=C:/mysql-cluster/data/ndb1
nodeid=2
NodeGroup=0

[ndbd]
hostname=ndb2
datadir=C:/mysql-cluster/data/ndb2
nodeid=3
NodeGroup=0

[ndbd]
hostname=ndb3
datadir=C:/mysql-cluster/data/ndb3
nodeid=4
NodeGroup=1

[ndbd]
hostname=ndb4
datadir=C:/mysql-cluster/data/ndb4
nodeid=5
NodeGroup=1
```

```
[mysqld]
hostname=mysqld
nodeid=50
```

De asemenea, am creat fișierul my.cnf pentru configurarea nodului SQL:

```
[mysqld]
ndbcluster
ndb-connectstring=mgmd
datadir=C:/mysql-cluster/data/mysqld
basedir=C:/mysql-cluster
```

5.2 Pornirea și Gestionarea Nodurilor

Pentru a facilita pornirea componentelor clusterului, am creat un set de scripturi batch în directorul C:\mysql-cluster\start. Aceste scripturi automatizează procesul de lansare a fiecărui nod în ordinea corectă și cu parametrii necesari:

- **start-mgmd.bat** - Pornește nodul de management. Acest script navigează în directorul bin și execută utilitarul ndb_mgmd.exe cu parametrii necesari:

  ```
  cd C:\mysql-cluster\bin
  ndb_mgmd.exe -f C:\mysql-cluster\config\config.ini
  --configdir=C:\mysql-cluster\config --ndb-nodeid=1
  ```

  Parametrul `-f` specifică locația fișierului de configurare, `--configdir` indică directorul care conține configurațiile, iar `--ndb-nodeid=1` asigură că acest proces primește ID-ul nodului 1, conform configurației.

- **start-mysqld.bat** - Pornește nodul SQL care oferă interfața pentru interogări:

  ```
  cd C:\mysql-cluster\bin
  mysqld --defaults-file=C:\mysql-cluster\my.cnf --ndbcluster
  --ndb-connectstring=mgmd --console
  ```

  Opțiunea `--defaults-file` specifică locația fișierului my.cnf, `--ndbcluster` activează motorul de stocare NDB, `--ndb-connectstring=mgmd` indică adresa nodului de management, iar `--console` afișează mesajele de ieșire în consola Windows pentru monitorizare facilă.

- **start-ndb1.bat**, **start-ndb2.bat**, **start-ndb3.bat**, **start-ndb4.bat**, - Aceste scripturi pornesc cele patru noduri de date care stochează efectiv informațiile:

  ```
  cd C:\mysql-cluster\bin
  ndbd.exe --ndb-connectstring=mgmd --ndb-nodeid=2
  ```

  ```
  cd C:\mysql-cluster\bin
  ndbd.exe --ndb-connectstring=mgmd --ndb-nodeid=3
  ```

  ```
  cd C:\mysql-cluster\bin
  ndbd.exe --ndb-connectstring=mgmd --ndb-nodeid=4
  ```

  ```
  cd C:\mysql-cluster\bin
  ndbd.exe --ndb-connectstring=mgmd --ndb-nodeid=5
  ```

Fiecare script pornește procesul ndbd.exe (Node Data Daemon) care gestionează un fragment de date. Parametrul `--ndb-connectstring` specifică adresa nodului de management pentru coordonare, iar `--ndb-nodeid` atribuie ID-ul corespunzător fiecărui nod, conform definiției din config.ini.

Ordinea corectă de pornire a componentelor este esențială: mai întâi nodul de management (mgmd), apoi nodurile de date (ndbd) și în final nodul SQL (mysqld). Această secvență asigură că nodul de management este disponibil pentru a coordona descoperirea și conectarea celorlalte noduri, iar nodurile de date sunt pregătite înainte ca serverul SQL să înceapă să primească interogări.

5.3 Testarea Funcționalității și Toleranței la Defecțiuni

Pentru a valida funcționalitatea clusterului și a demonstra toleranța la defecțiuni, am realizat o serie de teste practice care simulează scenarii reale de utilizare și eșec. Aceste teste verifică două aspecte esențiale: capacitatea sistemului de a gestiona operațiile obișnuite de bază de date și comportamentul în situații de defecțiune a nodurilor.

5.3.1 Pregătirea Mediului de Test

Prima etapă constă în conectarea la nodul SQL folosind clientul MySQL din linia de comandă:

```
C:\mysql-cluster\bin>mysql -u root -p
```

Această comandă stabilește o conexiune la serverul MySQL utilizând contul administratorului. Autentificarea cu utilizatorul root permite crearea și modificarea structurilor de baze de date, operațiuni necesare pentru teste.

5.3.2 Crearea și Popularea unei Baze de Date de Test

După autentificare, am creat o bază de date de test și o tabelă pentru stocarea informațiilor despre clienți:

```
CREATE DATABASE test_cluster;
USE test_cluster;
CREATE TABLE clienti (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nume VARCHAR(100),
    email VARCHAR(100)
) ENGINE=NDBCLUSTER;
```

Aspectul crucial în această definiție este specificarea motorului de stocare `ENGINE=NDBCLUSTER`. Această directivă instruiește MySQL să utilizeze motorul NDB pentru gestionarea tabelei, ceea ce înseamnă că:

- Datele vor fi distribuite automat între nodurile de date disponibile
- Sistemul va asigura replicarea conform parametrului `NoOfReplicas` din configurație
- Metadatele tabelei vor fi gestionate de nodul de management
- Operațiile de recuperare și redistribuire în caz de defecțiune vor fi automate

Pentru popularea tabelei, am inserat un prim rând de test:

```
INSERT INTO clienti (nume, email)
VALUES ('Ion Popescu', 'ion.popescu@example.com');
```

Verificarea datelor confirmă inserarea corectă:

```
SELECT * FROM clienti;
```

5.3.3 Monitorizarea Stării Clusterului

Pentru a obține o imagine de ansamblu asupra arhitecturii operaționale a clusterului, am utilizat utilitarul de management `ndb_mgm`:

```
C:\mysql-cluster\bin>ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to management server at localhost port 1186 (using cleartext)
Cluster Configuration
---------------------
[ndbd(NDB)]     4 node(s)
id=2    @127.0.0.1  (mysql-9.3.0 ndb-9.3.0, Nodegroup: 0)
id=3    @127.0.0.1  (mysql-9.3.0 ndb-9.3.0, Nodegroup: 0, *)
id=4    @127.0.0.1  (mysql-9.3.0 ndb-9.3.0, Nodegroup: 1)
id=5    @127.0.0.1  (mysql-9.3.0 ndb-9.3.0, Nodegroup: 1)
[ndb_mgmd(MGM)] 1 node(s)
id=1    @127.0.0.1  (mysql-9.3.0 ndb-9.3.0)
[mysqld(API)]   1 node(s)
id=50   @127.0.0.6  (mysql-9.3.0 ndb-9.3.0)
```

Ieșirea comenzii `show` oferă informații valoroase despre topologia clusterului:

- Există patru noduri de date (ndbd) organizate în două grupuri (Nodegroup 0 și 1)
- Un singur nod de management (ndb_mgmd) cu ID-ul 1 coordonează întregul cluster
- Un nod SQL (mysqld) cu ID-ul 50 furnizează interfața pentru interogări
- Simbolul asterisk (*) marchează nodul coordonator din cadrul unui grup de noduri
- Toate componentele rulează versiunea 9.3.0 a software-ului MySQL

Această structură asigură redundanța necesară, întrucât fiecare grup de noduri conține o copie completă a datelor, conform configurației.

5.3.4 Simularea Defecțiunii unui Nod

Pentru a testa toleranța la defecte, am simulat oprirea controlată a unui nod de date:

```
ndb_mgm> 4 stop
Node 4: Node shutdown initiated
Node 4 has shutdown.
```

Comanda `4 stop` oprește nodul cu ID-ul 4, care face parte din grupul de noduri 1. În acest moment, sistemul ar trebui să continue să funcționeze utilizând celelalte noduri disponibile. Pentru a verifica acest comportament, am inserat un nou rând în tabela de test:

```
INSERT INTO clienti (nume, email)
VALUES ('Maria Ionescu', 'maria.ionescu@example.com');
```

Operația s-a executat cu succes, după cum confirmă mesajul de răspuns:

```
Query OK, 1 row affected (0.008 sec)
```

Interogarea tabelei demonstrează că sistemul este perfect funcțional chiar și în absența unuia dintre nodurile de date:

```
mysql> SELECT * FROM clienti;
+----+--------------+--------------------------+
| id | nume         | email                    |
+----+--------------+--------------------------+
|  2 | Maria Ionescu | maria.ionescu@example.com |
|  1 | Ion Popescu   | ion.popescu@example.com   |
+----+--------------+--------------------------+
2 rows in set (0.007 sec)
```

Acest comportament demonstrează arhitectura tolerantă la defecțiuni a MySQL Cluster, care permite continuarea neîntreruptă a operațiunilor de bază de date chiar și în cazul eșecului unui nod. Observăm că:

- Timpul de execuție al operațiilor rămâne similar (0.008 secunde pentru INSERT, 0.007 secunde pentru SELECT)
- Nu s-a produs nicio întrerupere perceptibilă a serviciului
- Datele sunt complet accesibile și consistente

5.3.5 Restaurarea Clusterului la Capacitate Completă

După testarea comportamentului în condiții de defecțiune, am repornit nodul oprit utilizând scriptul corespunzător. Sistemul de management detectează și confirmă revenirea nodului:

```
ndb_mgm> Node 4: Started (version 9.3.0)
```

Verificarea configurației clusterului confirmă restaurarea completă a arhitecturii:

```
show
Cluster Configuration
---------------------9
[ndbd(NDB)]     4 node(s)
id=2    @127.0.0.1  (mysql-9.3.0 ndb-9.3.0, Nodegroup: 0)
id=3    @127.0.0.1  (mysql-9.3.0 ndb-9.3.0, Nodegroup: 0, *)
id=4    @127.0.0.1  (mysql-9.3.0 ndb-9.3.0, Nodegroup: 1)
id=5    @127.0.0.1  (mysql-9.3.0 ndb-9.3.0, Nodegroup: 1)
[ndb_mgmd(MGM)] 1 node(s)
id=1    @127.0.0.1  (mysql-9.3.0 ndb-9.3.0)
[mysqld(API)]   1 node(s)
id=50   @127.0.0.6  (mysql-9.3.0 ndb-9.3.0)
```

La repornire, nodul reintegrat în cluster sincronizează automat toate modificările efectuate în timpul absenței sale. Acest proces de recuperare demonstrează mecanismele robuste de auto-vindecare implementate în MySQL Cluster, care asigură atât continuitatea serviciilor cât și coerența datelor în scenarii de defecțiune și recuperare.

5.3.6 Concluzii ale Testelor

Testele efectuate demonstrează următoarele aspecte esențiale ale MySQL Cluster:

- **Transparența defecțiunilor** - Aplicațiile client nu detectează eșecul nodurilor individuale
- **Disponibilitate ridicată** - Serviciul rămâne 100% funcțional în timpul defecțiunilor parțiale
- **Recuperare automată** - Nodurile repornite sunt reintegrate fără intervenție manuală
- **Consistența datelor** - Informațiile rămân coerente și complete în toate fazele testului

Aceste caracteristici confirmă adecvarea MySQL Cluster pentru scenarii critice unde întreruperile serviciului și pierderea datelor sunt inacceptabile, validând astfel principiile teoretice discutate în secțiunile anterioare ale lucrării.

6 Concluzii

MySQL Cluster reprezintă o evoluție semnificativă în domeniul bazelor de date distribuite, îmbinând disponibilitatea ridicată cu consistența strictă a datelor. Prin eliminarea punctelor unice de eșec și automatizarea procedurilor de recuperare, această tehnologie se potrivește perfect aplicațiilor critice pentru afaceri, unde perioadele de nefuncționare sunt inacceptabile. Cercetările viitoare ar putea explora integrarea cu tehnologii moderne precum containerele Docker sau orchestrarea Kubernetes pentru o gestionare și mai flexibilă a infrastructurii.

Bibliografie

[1] Özsu, M. T., & Valduriez, P. (2020). *Principles of Distributed Database Systems*. Springer Nature.

[2] Ashvin Choudhary, *Understanding CAP Theorem: Basics and Real-World Examples*, Medium, 17 mai 2023. Disponibil online: https://ashvinchoudhary.medium.com/understanding-cap-theorem-real-world-examples-and-databases-d1ce0d807dca. Accesat la 14 mai 2025.

[3] Oracle Corporation. (2025). *MySQL Cluster Overview*. MySQL 9.3.0 Reference Manual. https://dev.mysql.com/doc/mysql-cluster-manager/1.4/en/mcm-overview.html

[4] Oracle Corporation. (2025). *MySQL Cluster Management Node*. MySQL 9.3.0 Reference Manual. https://dev.mysql.com/doc/refman/9.0/en/mysql-cluster-overview.html

[5] Oracle Corporation. (2025). *MySQL Cluster Replication*. MySQL 9.3.0 Reference Manual. https://dev.mysql.com/doc/refman/9.0/en/mysql-cluster-replication.html