Baze de date

anul 1, semestrul 2

Curs 3

- De ce este nevoie de *caching*? Avantaje și cazuri de utilizare.
- Medii de stocare pentru *caching*.
- Strategii de *caching*.
- Exemple de baze de date IN-Memory, exemple de utilizare.
- Exemplu proiect ERD

CACHING

STOCAREA DATELOR FRECVENT UTILIZATE
COPIEREA ACESTORA ÎN MEDII DE STOCARE
CARE PERMIT ACCES RAPID.

Caching *avantaje*

- **Performanță** ridicată, rezolvarea rapidă a interogărilor.
- **Workload redus** pentru baza de date, prin eliminarea unor sarcini repetitive.
- **Availability**: date accesibile chiar dacă serverele de bd sunt temporar inaccesibile.
- **Separation of concerns** -> *load-balancing* eficient și costuri reduse (server load).

Caching *exemple de cazuri de utilizare*

- **Comerț electronic**: accesarea rapidă a produselor aflate în promoție sau a produselor apreciate.
- **Gamining**: clasamente, features și assets, date despre utilizatori.
- **Date de sesiune**: profil utilizator, browsing history.
- **Streaming video**: pentru a reduce buffering-ul și a îmbunătăți experiența utilizatorului.
- **Aplicații mobile**: datele aplicației, preferințele de limbă, categoriile accesate frecvent.

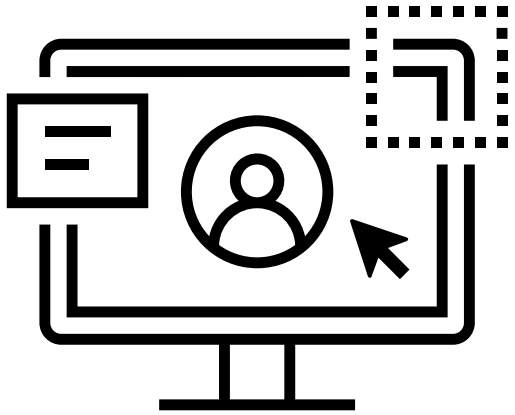
Caching *medii de stocare*

- *Baze de date in-memory*:
  - baze de date care stochează toate datele în memoria RAM.
  - timpi de acces mult mai mici comparativ cu timpii pentru accesare și stocarea pe disc.
  - Pot fi relaționale sau NoSql dar nu asigură ACI**D** (durabilitate).
  - Pot fi utilizate în ecosisteme ***cloud*** aplicații care necesită performanță ridicată și latență scăzută.

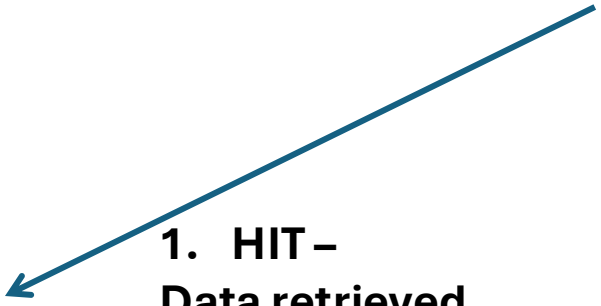
Caching *medii de stocare*

- *Caching în cloud*:
  - **Edge caching**: datele sunt stocate pe servere de *margine* (*edge servers*) situate în apropierea utilizatorilor finali. (obiective turistice, IoT, aplicații bancare, comenzi ale clientilor pe zone)
  - **CDN (content delivery network)** sunt rețele distribuite de data centre și servere proxy care stochează conținutul pe baza locației geografice.
- *Direct în aplicație*:
  - Serviciile backend pot stoca datele direct în aplicație, stocare locală.

Caching strategii READ *Cache aside*



WEB APP/
MOBILE APP/
BACKEND

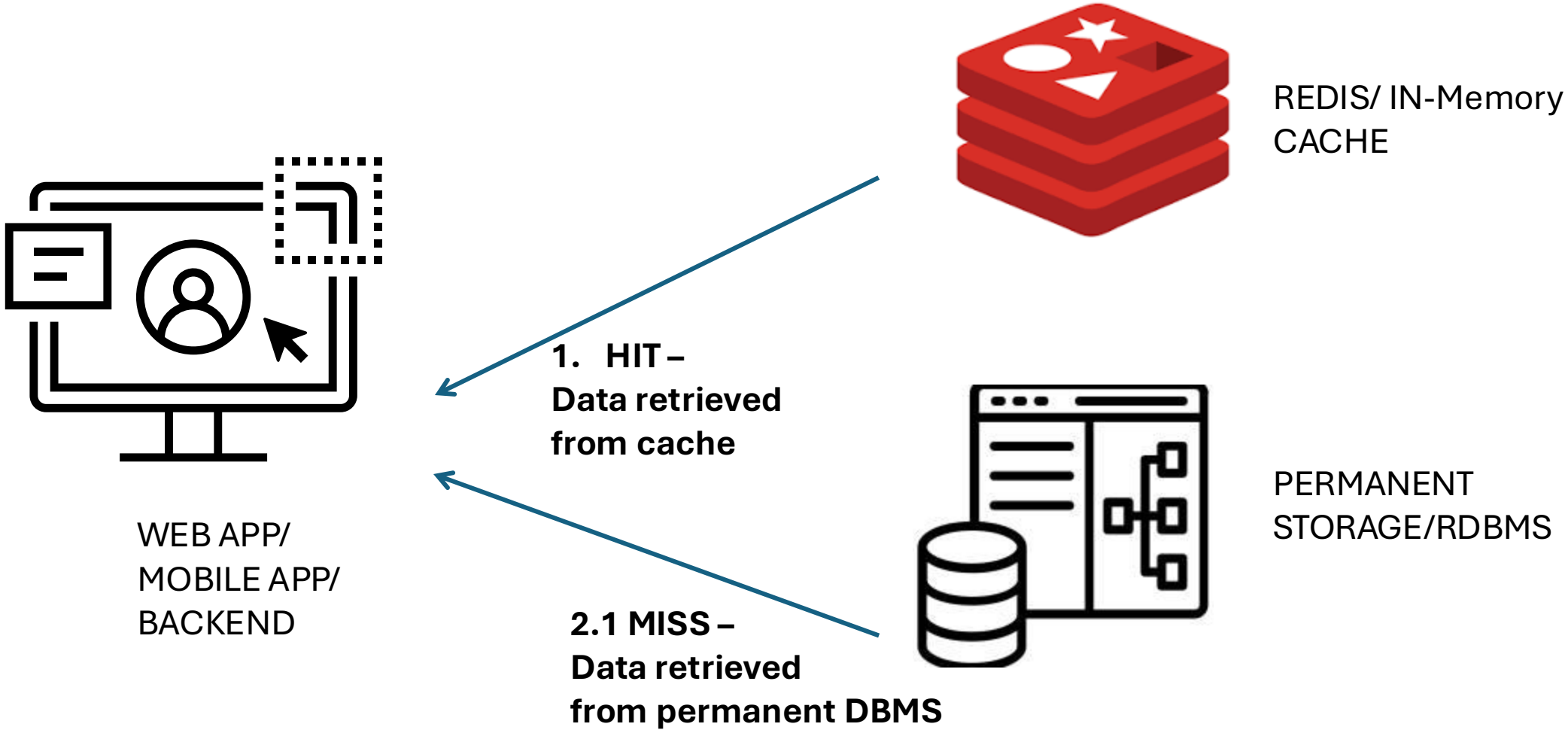


1. HIT – Data retrieved from cache

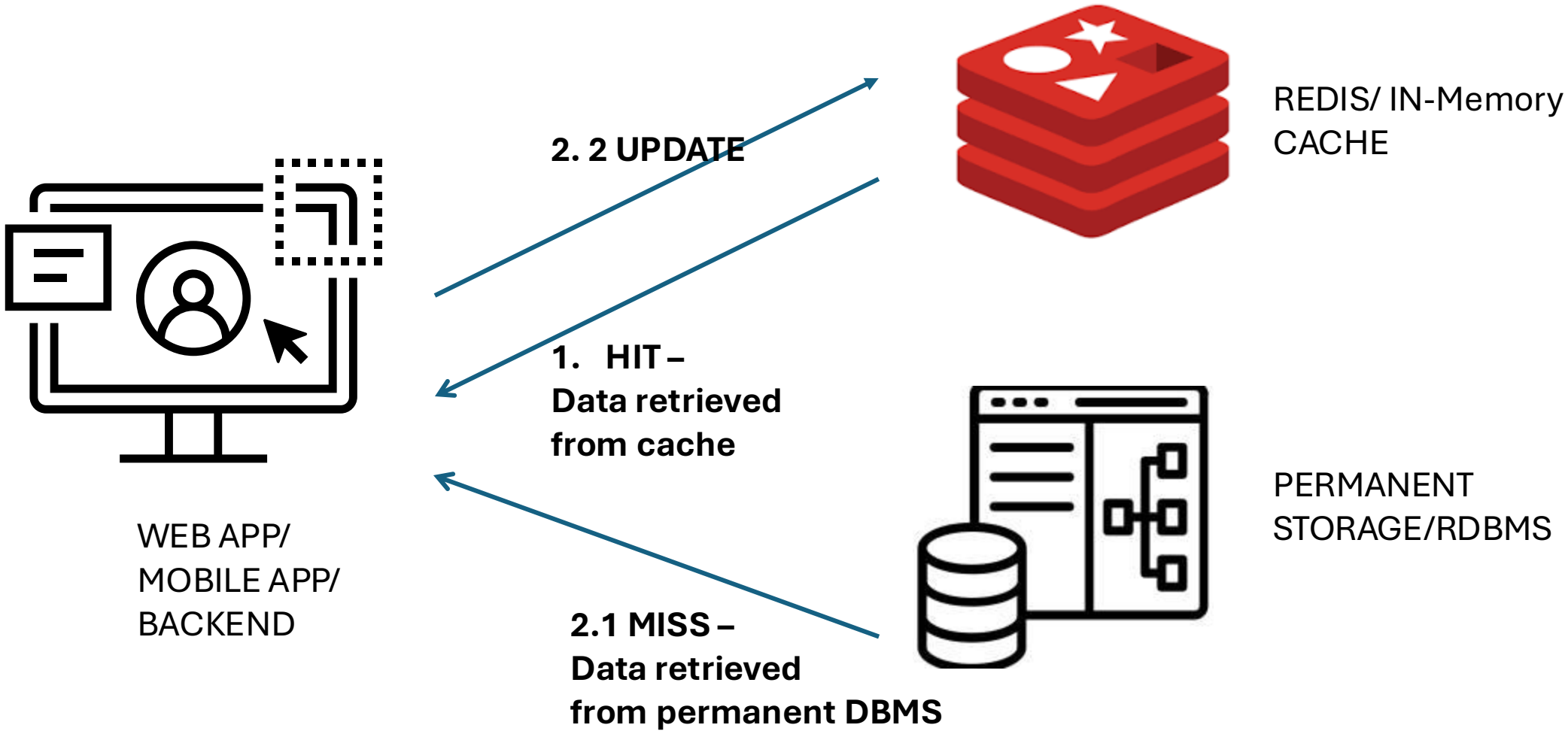
REDIS/ IN-Memory CACHE

PERMANENT STORAGE/RDBMS

Caching strategii READ *Cache aside*



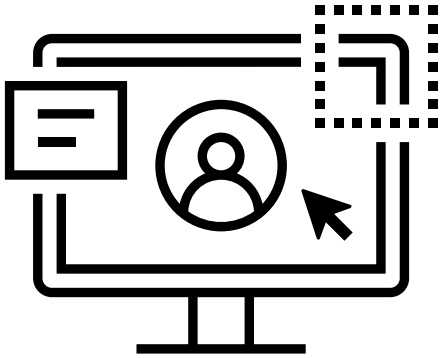
Caching strategii READ *Cache aside*



Caching strategii pentru READ

- ***Cache aside***:
  - *Lazy loading*: datele sunt stocate în cache atunci când este necesar.
  - Aplicația este responsabilă pentru actualizarea bazei de date cache.
  - *Resilience*: Dacă nu este disponibil cache-ul atunci datele sunt obținute din baza de date permanentă.
  - A**C**ID Consistency: Datele citite din cache pot fi incorecte.
  - detalii produse, platforme de streaming.

Caching strategii READ *Read-Through*



WEB APP/
MOBILE APP/
BACKEND

1. REQUEST

3. RESPONSE

CACHE

2a. HIT – Data retrieved from cache

2b. 2 UPDATE

2b.1 MISS – Data retrieved from permanent DBMS

PERMANENT STORAGE/RDBMS

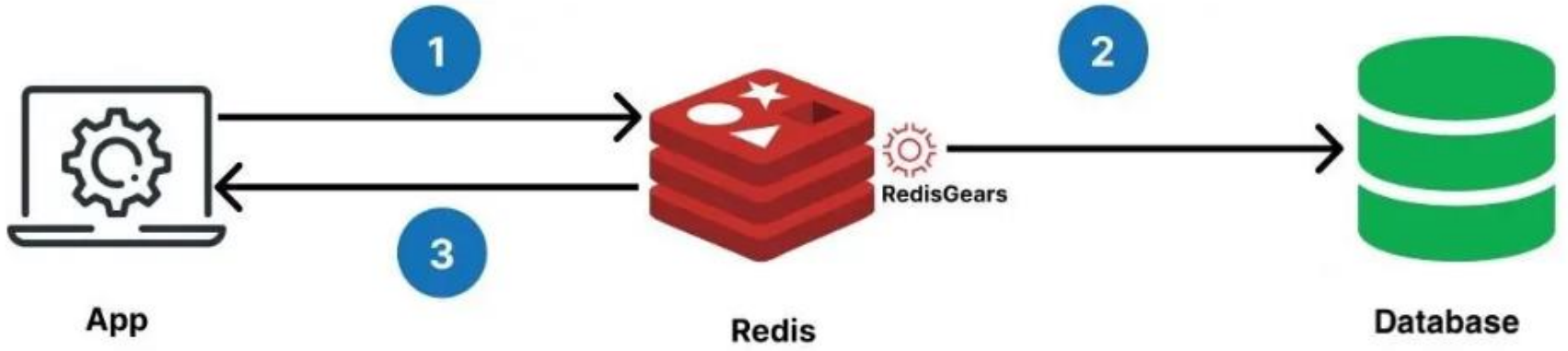
REDIS/ IN-Memory CACHE

Caching strategii READ

- ***Read-Through***:
  - *Cache layer* este responsabilă pentru actualizarea bazei de date cache.
  - *Resilience*: Dacă nu este disponibil cache-ul atunci datele nu pot fi obținute.
  - A**C**ID Consistency: Datele citite din cache sunt consistente dacă este folosită strategia de *write* potrivită (***Write-Through***).
  - gaming, asigurari, profil utilizator, date pacienti.

Caching strategii READ

- ***RedisGears***:
  - engine de procesare a datelor pentru Redis
  - permite rularea de funcții personalizate (Python, C) direct în Redis (*serverless*).
  - acest lucru ajută la prelucrarea și analiza datelor fără a muta informațiile între Redis și o altă bază de date, *event-driven*



Caching strategii WRITE

- ***Write-Through:***
  - Datele sunt scrise simultan atât în cache, cât și în baza de date principală în momentul în care sunt actualizate.
  - Asigură consistența datelor între cache și stocarea principală, deoarece datele sunt întotdeauna actualizate în ambele locații.
  - Poate avea o performanță mai scăzută comparativ cu alte metode de caching, deoarece fiecare operațiune de scriere implică două scrieri (în cache și în stocarea principală).

Caching strategii WRITE

- ***Write-Back:***
  - Datele sunt scrise inițial doar în cache, iar actualizarea stocării principale se face ulterior, într-un moment ulterior sau când datele sunt eliminate din cache.
  - Poate îmbunătăți performanța, deoarece operațiunile de scriere sunt mai rapide (se fac doar în cache inițial).
  - Există riscul de pierdere a datelor în cazul unei defecțiuni înainte ca datele să fie scrise în stocarea principală.

Caching strategii WRITE

- ***Write-Around:***
  - Datele sunt scrise direct în stocarea principală. Cache-ul este actualizat doar atunci când datele sunt citite ulterior.
  - Avantaje: Reduce încărcarea pe cache pentru operațiunile de scriere, ceea ce poate fi benefic pentru aplicațiile cu un număr mare de scrieri.
  - Dezavantaje: Poate duce la o performanță mai scăzută pentru citirile ulterioare, deoarece datele nu sunt disponibile imediat în cache și trebuie citite din stocarea principală.

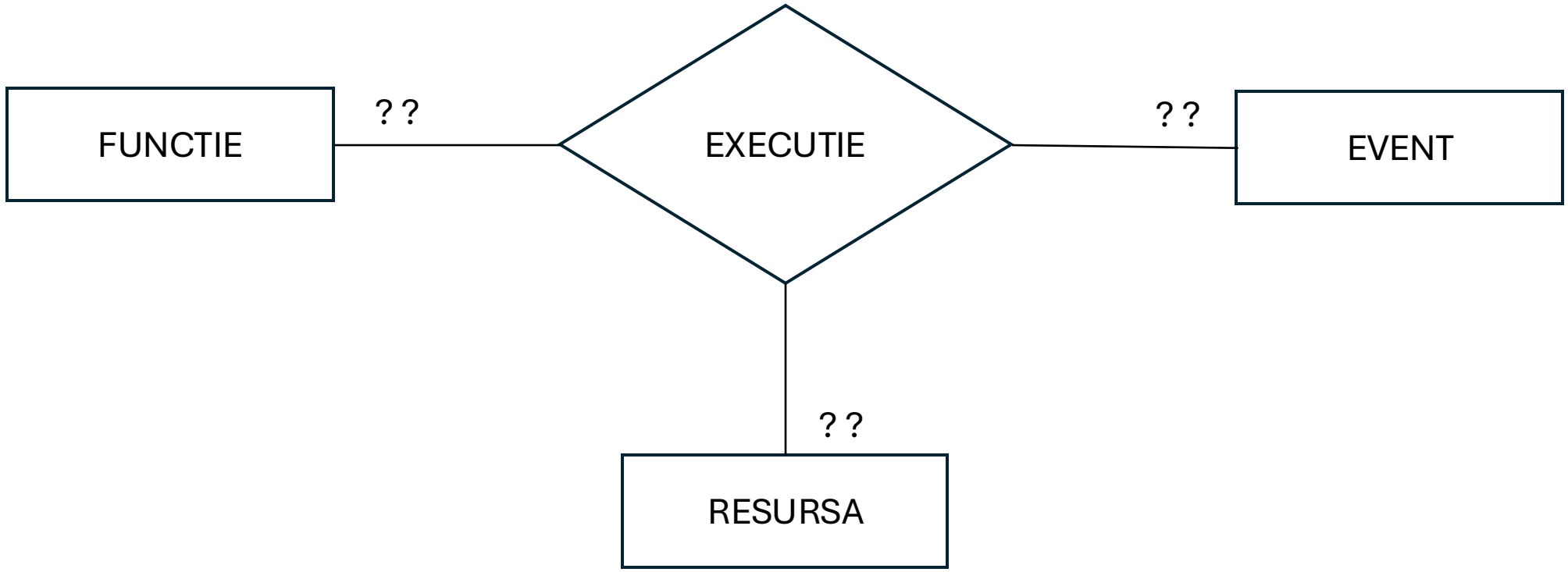
Caching *Exemple de baze de date in memory*

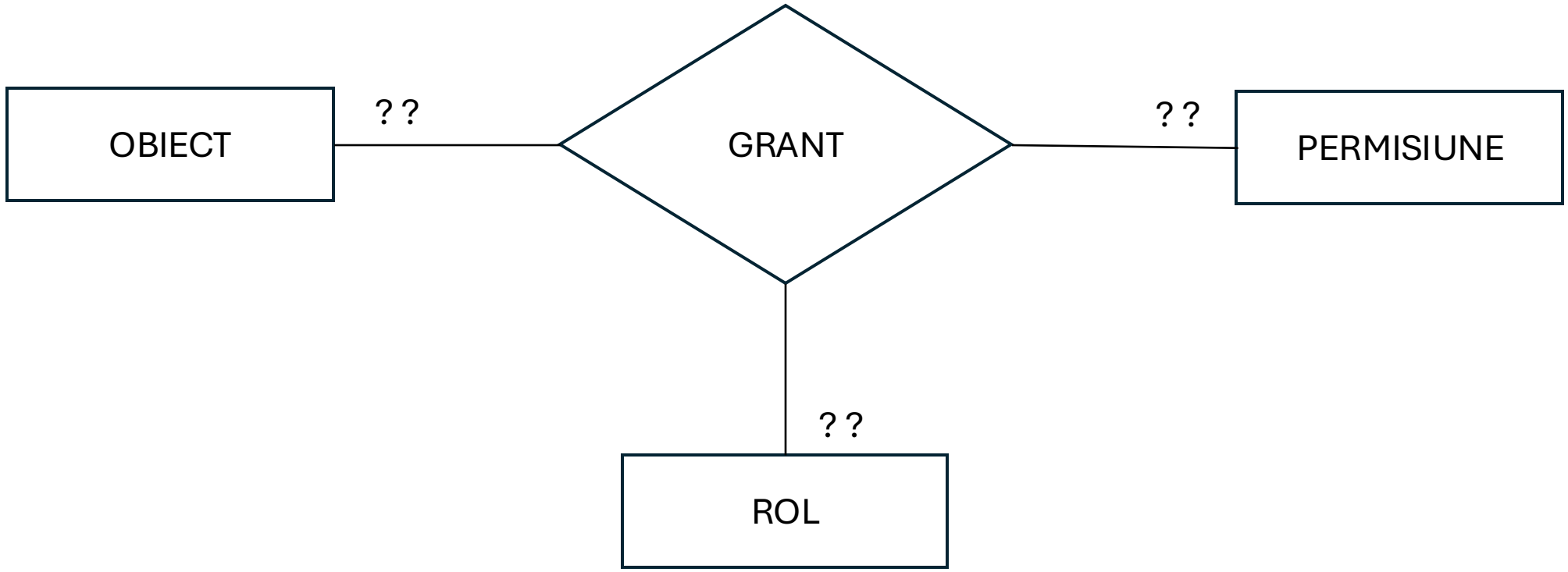
- ***Redis***:
  - Bază de date NoSQL open-source, utilizată pentru caching, mesagerie și stocarea datelor în timp real.
  - Caracteristici: Suportă structuri de date precum stringuri, liste, seturi, hărți hash, bitmaps, streamuri și indici spațiali.
  - Utilizare: Este folosită de companii precum Twitter, Airbnb, Adobe, Hulu, Amazon și OpenAI.

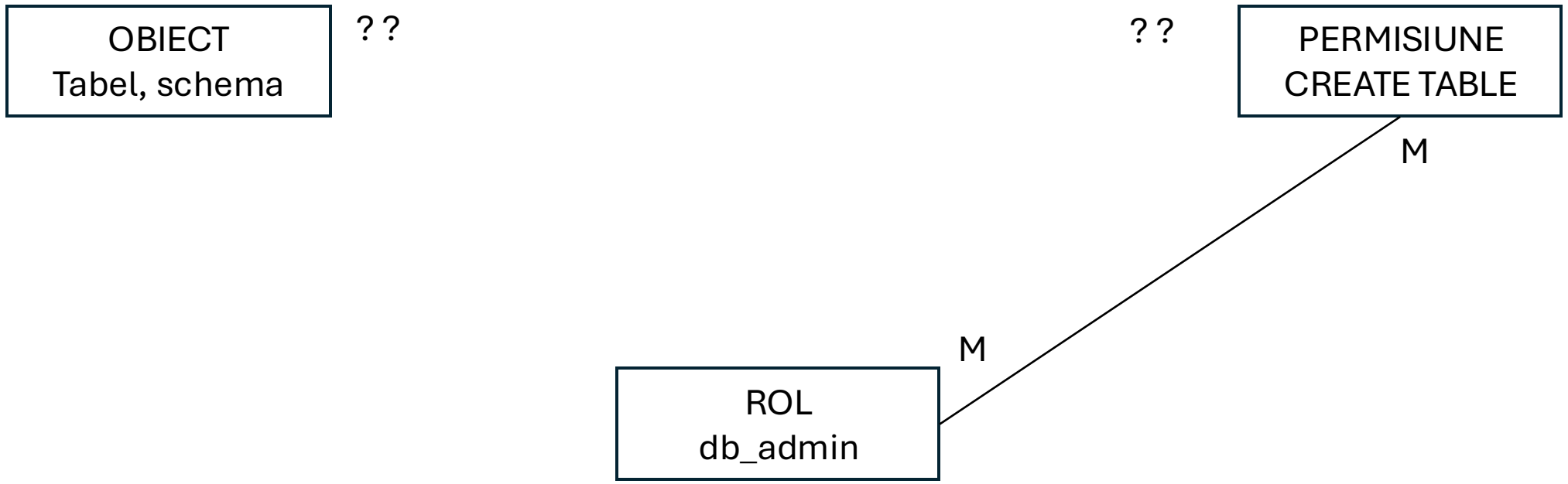
Caching *Exemple de baze de date in memory*

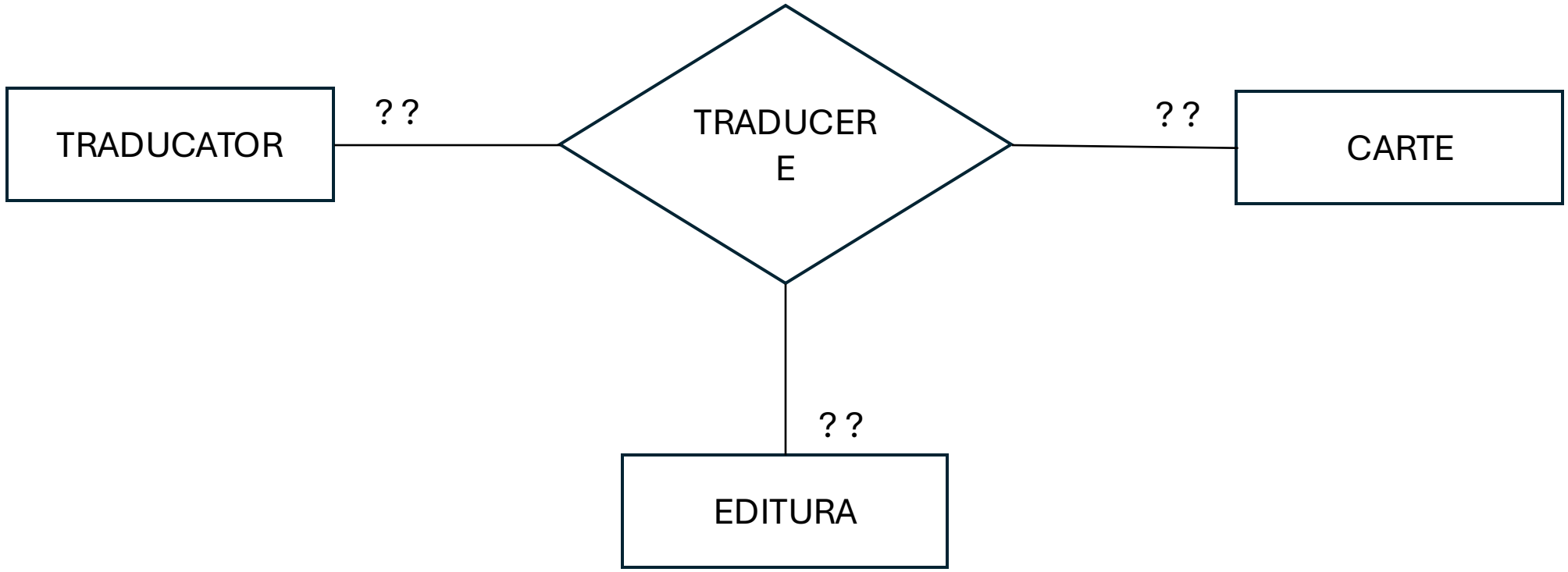
- ***Apache Ignite***:
  - Bază de date distribuită de tip key-value.
  - Caracteristici: Oferă suport pentru SQL, key-value și procesare de date, și poate opera în mod pur in-memory sau cu persistență.
  - Utilizare: Este utilizată pentru aplicații care necesită performanță ridicată și scalabilitate, cum ar fi calcule necesare în ML.

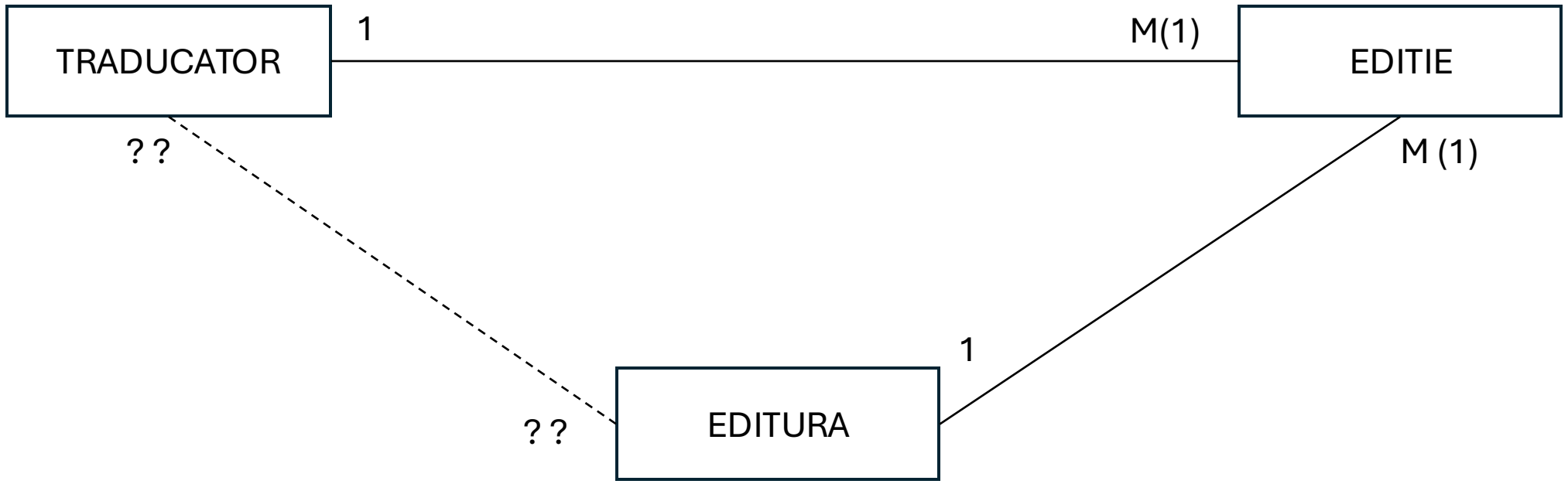
- ***Memcache***
  - Bază de date distribuită de tip key-value.
  - Caracteristici: Design simplu, date arbitrare de dimensiune redusă (stringuri, obiecte) rezultate din apeluri de baze de date, apeluri API.

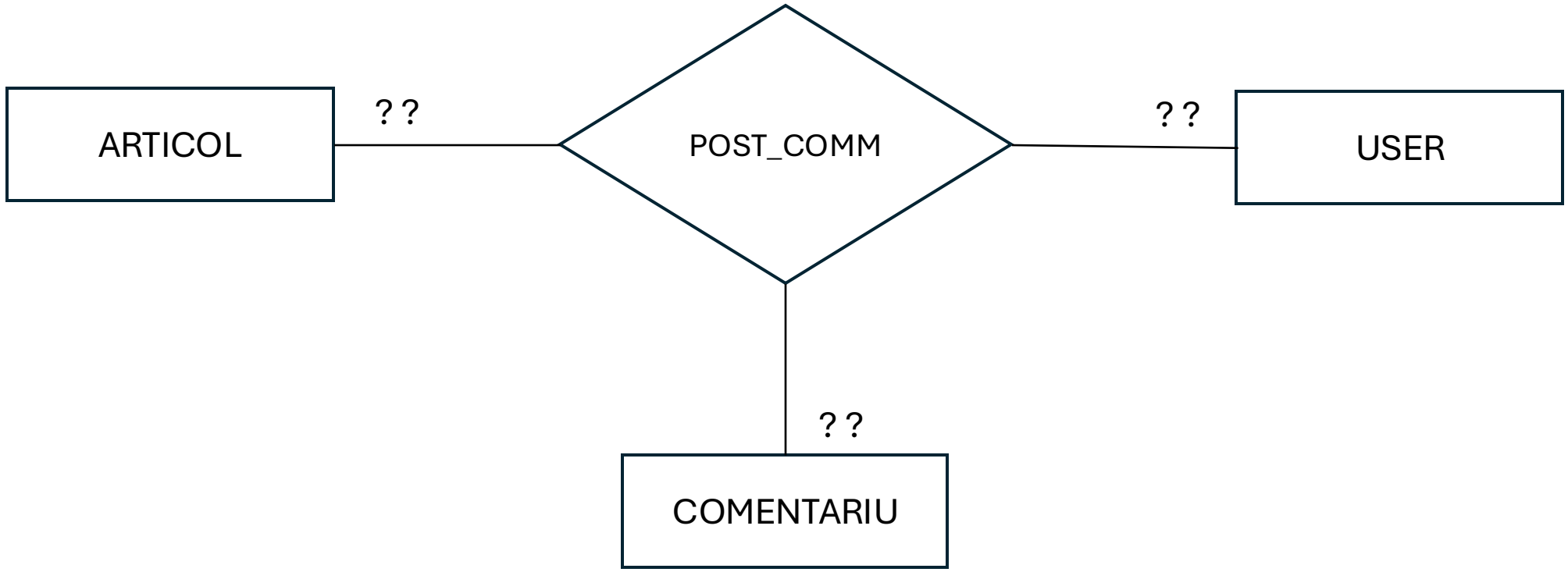

FUNCTIE
? ?
EXECUTIE
? ?
EVENT
? ?
RESURSA

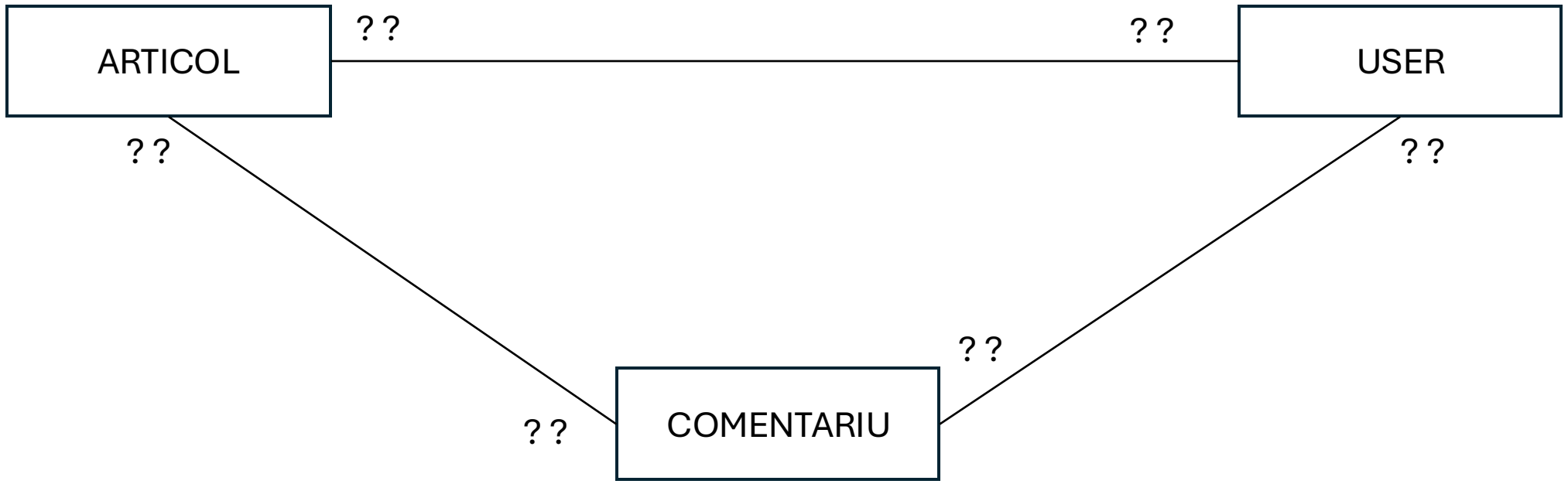

OBIECT
? ?
GRANT
? ?
PERMISIUNE
? ?
ROL

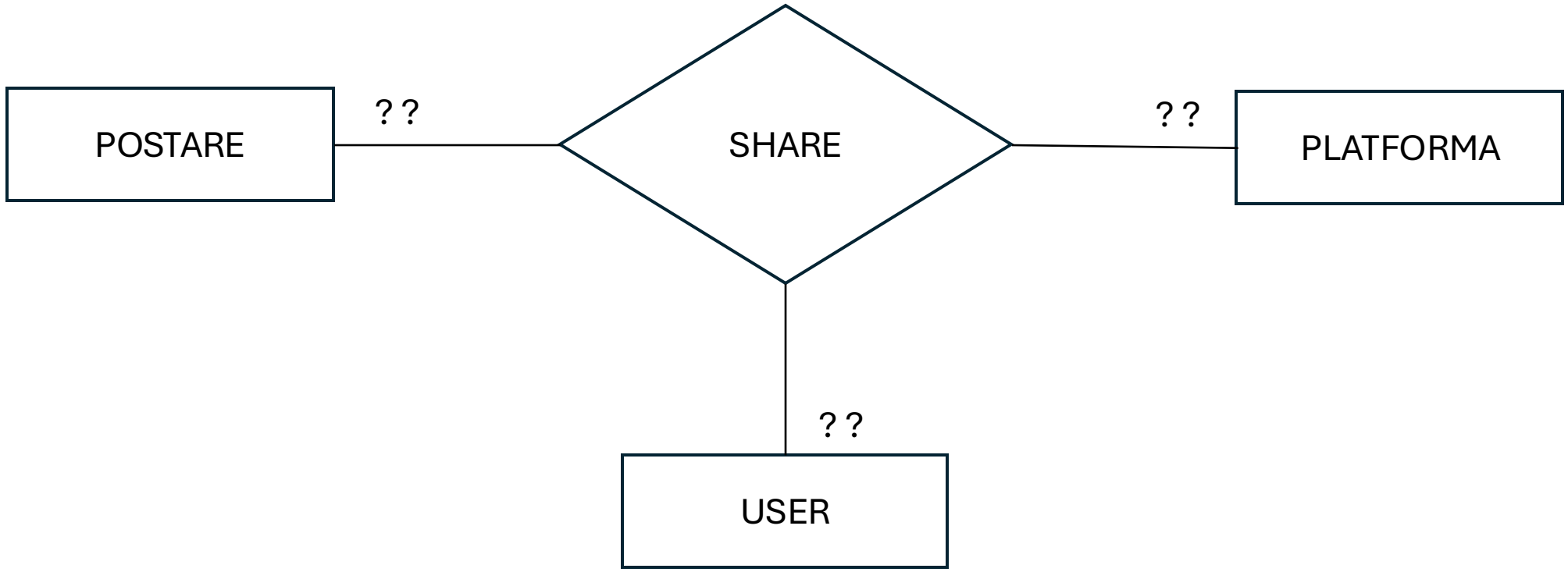

OBIECT
Tabel, schema
? ?
? ?
PERMISIUNE
CREATE TABLE
M
M
ROL
db_admin

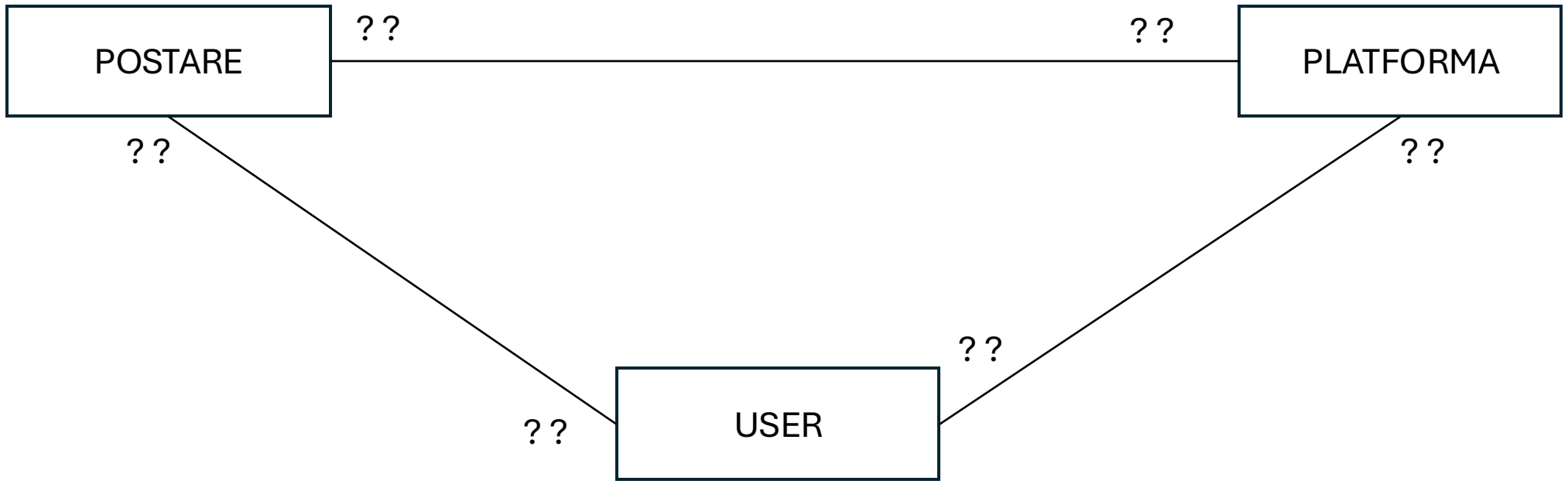

TRADUCATOR
? ?
TRADUCER
E
? ?
CARTE
? ?
EDITURA


TRADUCATOR
1
M(1)
EDITIE
? ?
M (1)
1
? ?
EDITURA


ARTICOL
? ?
POST_COMM
? ?
USER
? ?
COMENTARIU


ARTICOL
? ?
? ?
USER
? ?
? ?
? ?
? ?
COMENTARIU


POSTARE
? ?
SHARE
? ?
PLATFORMA
? ?
USER


POSTARE
? ?
? ?
PLATFORMA
? ?
? ?
? ?
? ?
USER

- https://blogs.oracle.com/database/post/introducing-oracle-true-cache
- https://www.dragonflydb.io/guides/in-memory-databases
- https://redis.io/learn/howtos/solutions/caching-architecture/write-through