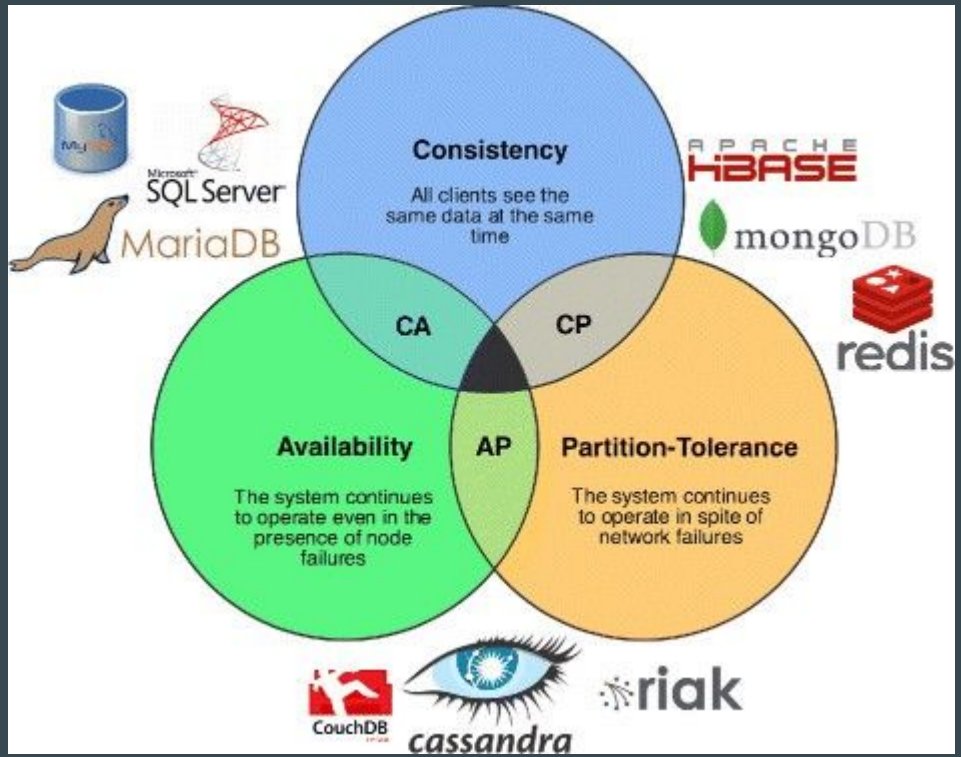
Baze de date

...

Teorema CAP



- Putem alege doar 2 din cele 3 calitati
- Combinatiile alese rezulta in tehnologii cu use-case-uri specifice
- Prin urmare, alegem tehnologia in functie de requirement-urile tehnice ale proiectului, si nu invers.

ACID

1. **Atomicity** : Each database transaction must completely succeed or completely fail. Partial success is not allowed. If the transaction includes several “steps” (either read or write) and one or more fails, the transaction will fail and the database will return to the state it was before starting the transaction.

2. **Consistency** : During the database transaction, the RDBMS progresses from one valid state to another. The state is never invalid. This means that if the database is replicated across several nodes (copies of the database), all nodes will hold the same information.

3. **Isolation** : The client’s database transaction must occur in isolation from other clients attempting to transact with the RDBMS. This means that if a transaction is taking place, other users will have to wait until it ends.

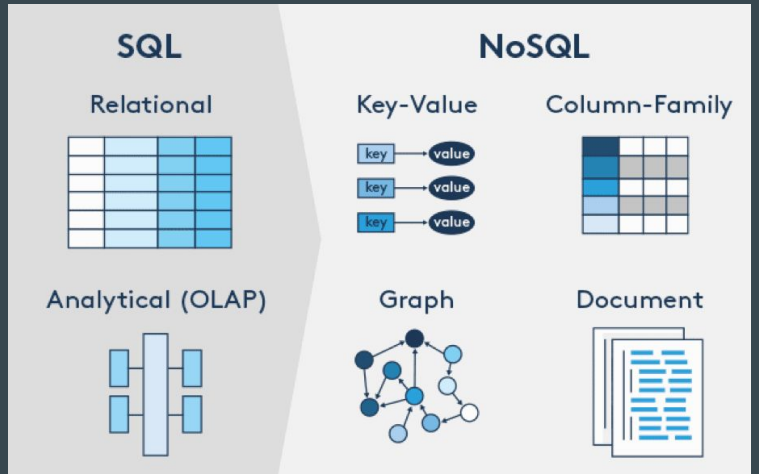
4. **Durability** : The data operation that was part of the transaction must be reflected in nonvolatile storage (computer memory that can retrieve stored information even when not powered — like a hard disk) and persist after the transaction successfully completes. Transaction failures cannot leave the data in a partially committed state.

SQL ~ Relational Data Model

- RDBMS
- Scalabilitate ~ verticala
- Schema predefinita / stabila
- Modeleaza bine date in principal relationale
- Query-uri complexe

NoSQL ~ Document Data Model

- Distributed dbms
- Scalabilitate ~ orizontala
- Schema dinamica
- Modeleaza bine date de tip document / big-data
- Se preteaza mai bine la query-uri simple



ORM: Object-relational mapping

- Layer de abstractie intre business logic si mediul de persistare a datelor (db)
- Motivatie:
  - Nu vrem sa avem query-uri SQL/altceva amestecate cu restul codului (Single Responsability Principle, DRY)
  - Nu vrem sa repetam acelasi query in multiple locuri in cod (DRY)
  - Vrem sa avem abilitatea de a lucra la nivel abstract cu obiecte
  - Vrem sa avem functionalitate cat mai rapid pentru prototipare (RAD)
  - Nu scaleaza intotdeauna foarte bine in raport cu complexitatea proiectului, dar in majoritatea cazurilor prezinta un punct bun de inceput.

ORMs: Active Record vs Data Mapper pattern

- Clasele de tip Model primesc prin inheritance functionalitate de manipulare a DB

```php
$user = new User();
$user->username = "philipbrown";
$user->save();
```

- Clasele de tip model nu stiu nimic despre constructie de query-uri
- Un serviciu de tip EntityManager se ocupa de layer de persistenta direct

```php
$user = new User();
$user->username = "philipbrown";
EntityManager::persist($user);
```

Bonus: Repository Pattern

- Un layer special in care query-urile sunt permise, in restul codebase-ului lucrandu-se doar cu obiecte dumb / containere de date.

Best Practices pt lucrul cu DB

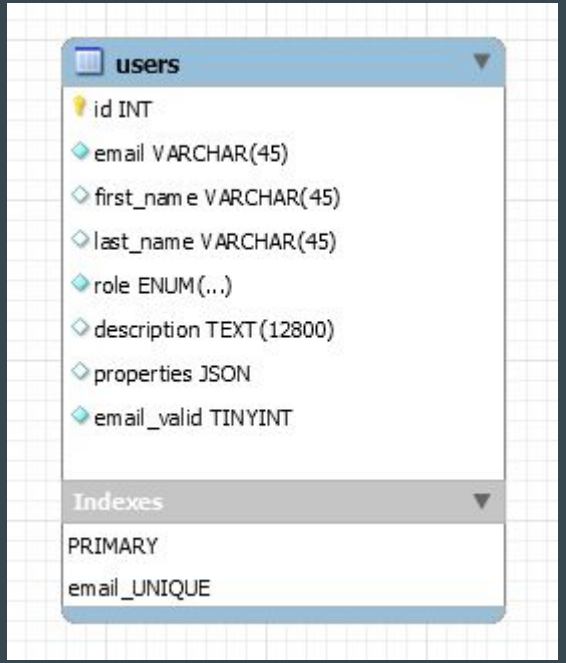
- DB schema as code
- Schimbari incrementale, care se dezvolta odata cu codul, in mod agil
- Date de test pentru medii de dezvoltare locale
- Query-uri des intalnite extrase intr-un layer de acces de date ~ repository
- Tooling pt a automatiza procesul de refresh al DB in timp util

Baze de date II

...

Modelarea unui DB Schema - tipuri de date, optiuni relevante

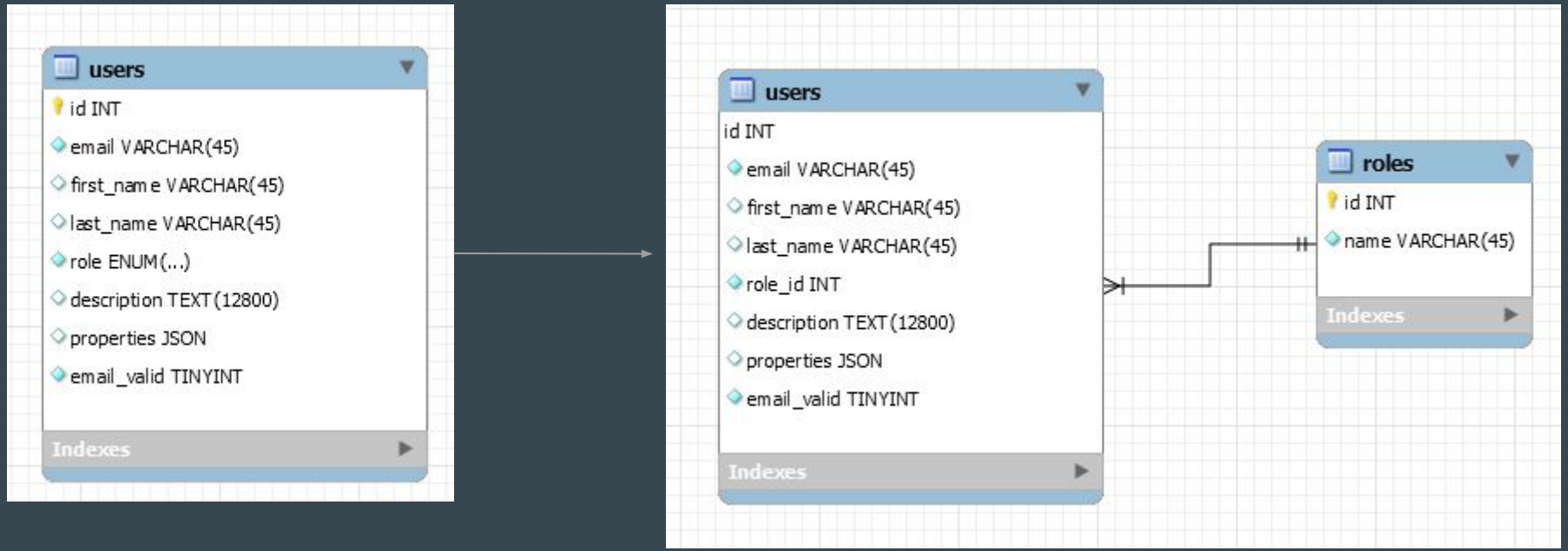
- Tipuri de date
  - Numeric: INT, BIGINT, FLOAT
  - DateTime: DATE, TIME, DATETIME
  - String: VARCHAR, BLOB, TEXT, ENUM
  - Misc: JSON, aplicatii pt GIS
- Optiuni la nivel de coloana
  - Primary Key (index)
  - Auto increment
  - Nullable
  - Unique (index)
  - Default



Modelarea unui DB Schema - indexing, foreign keys, constraints

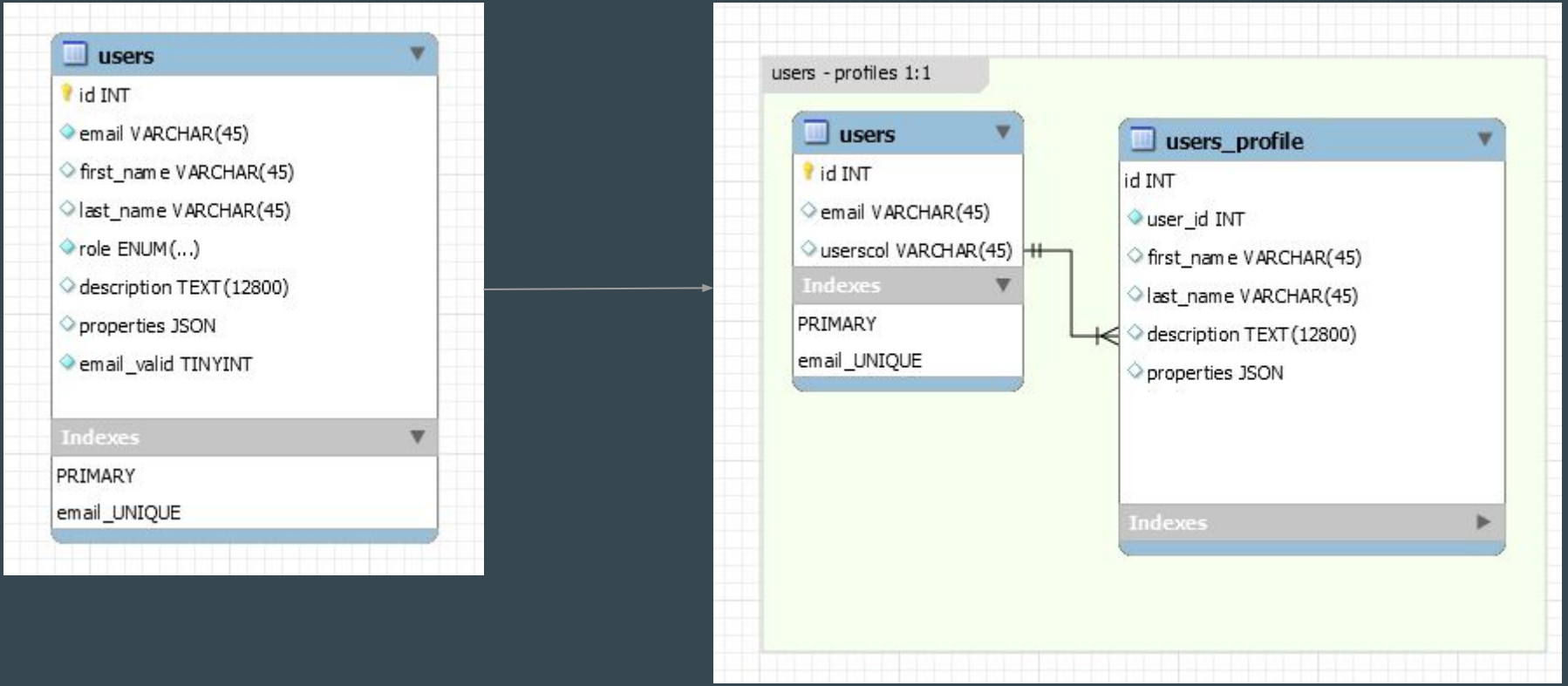
- Primary Key
- Unique
- Index (+ composite)
- Foreign Key (+on delete, CASCADE)

Modelarea unui DB Schema - normalizare

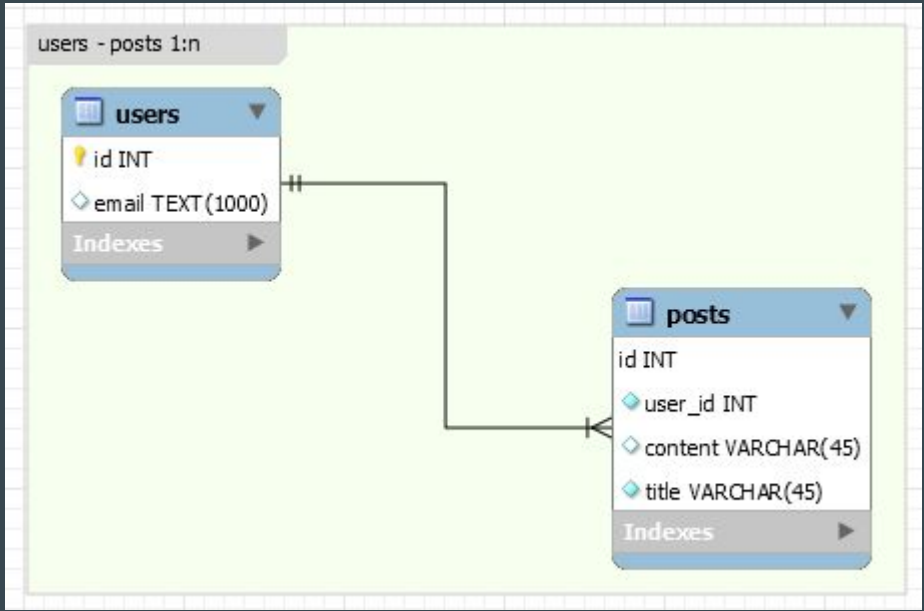


Exemplu de normalizare, unde ENUM, care ar fi fost redundant, primeste propria tabela

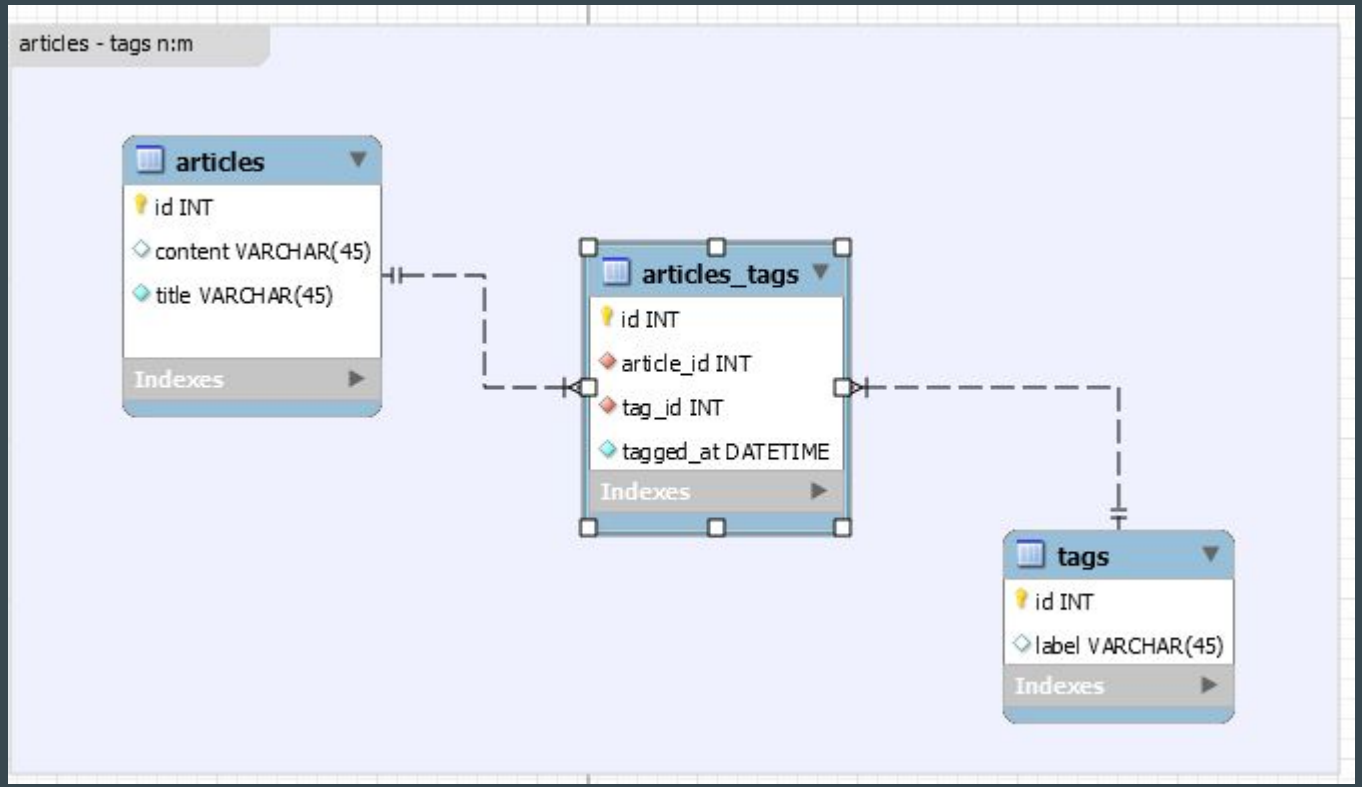
Relatii la nivel de ORM: One to One (1:1)



Relatii la nivel de ORM: One to Many (1:n)



Relatii la nivel de ORM: Many to Many (n:m)



Relatii la nivel de ORM: Polymorphic (1:1)