**Document Structure Analyse**

**1. File met nodes in en property type van de node, relaties worden als array bewaard als property van de node**

* Voordelen
  + **Simpel model:** 
    - Alles staat in één collectie, makkelijk te importeren/exporteren.
  + **Makkelijk op te zetten:**
    - Geen complexe schema’s, gewoon 1 JSON per node.
  + **Flexibiliteit:** 
    - Je kan nieuwe types en relaties toevoegen zonder schema migraties.
  + **Goed voor prototyping:**
    - Vooral handig als je snel iets met een webapp wil testen.
* Nadelen
  + **Query’s worden complex:**
    - Je moet veel joins in de applicatie laag doen (want Mongo heeft een echte joins).
    - Relaties op meerdere hops zijn traag en lastig.
  + **Redundantie of inconsistentie:**
    - Als je een relatie in Node A opslaat, moet je dit eigenlijk ook in Node B spiegelen, zorgt voor kans op inconsistentie.
  + **Performance issues bij groei:**
    - Zodra er veel relaties zijn, worden arrays groot, updates en query’s worden trager.
  + **Geen echte graph-functionaliteit meer:**
    - Traversals die in een graph DB zijn kunnen hier niet.

**Afbeelding met tekst, schermopname, menu

Automatisch gegenereerde beschrijving**

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving**2. Document maken per type node, relaties bewaren als array in property van de record**

* Voordelen
  + **Betere organisatie:**
    - Data is opgesplitst per type, waardoor documenten per collectie homogener zijn.
  + **Efficiënter zoeken:**
    - Query’s per node-type zijn sneller.
  + **Minder “type”-property nodig:** 
    - Je weet al uit de collectie wat voor node het is, scheelt data & query’s.
  + **Makkelijker schema’s opstellen**:
    - Voorbeeld validatie per collectie.
  + **Goed voor API-structuur:**
    - In je webapp kan je endpoints netjes mappen op collecties.
* Nadelen
  + **Relaties blijven arrays**
    - Je hebt nog steeds duplicatie en kans op inconsistenties (Node 1 zegt dat hij verbonden is met Node B, maar Node B heeft dat niet correct gespiegeld).
  + **Cross-collection query’s lastiger**
    - Als je een query wil zoals “geef alle parts van exo X inclusief joints” zul je meerdere query’s moeten doen en data zelf samenvoegen.
  + **Traversals blijven moeilijk**
    - Diepere graph query’s (2 of meer hops) zijn lastig zonder extra logica in je applicatie.
  + **Moeilijker bij dynamische relaties**
    - Als relaties heel gevarieerd zijn (veel soorten edges), kan het beheer ingewikkeld worden.

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, scherm

Automatisch gegenereerde beschrijving

**Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving3. Nodes en verschillende relaties in aparte collecties bewaren**

* Voordelen
  + **Duidelijke semantiek:** 
    - Elke collectie vertegenwoordigt één type relaties, waardoor het heel leesbaar is en logische aanvoelt.
  + **Query’s per relatie-type zijn snel:** 
    - Als je bijvoorbeeld alle *HAS\_AIM* relaties nodig hebt, zit je meteen in de juiste collectie zonder filtering.
  + **Indexering** **eenvoudig**:
    - Je kan makkelijk indexen op relatie binnen een specifieke collectie.
  + **Schaalbarer dan arrays:** 
    - Relaties staan los van Nodes, dus geen enorme arrays die steeds groter worden
  + **Past bij een webapp die endpoints per relatie-type wil:**
    - Bijvoorbeeld /belongs\_to, /has\_aim
* Nadelen
  + **Fragmentatie**
    - Relaties zijn verspreid over meerdere collecties, wat query’s lastiger maakt als je “alle relaties van een node” wil, moet je dan door alle relatie-collecties heen zoeken.
  + **Extra complexiteit**
    - Je backend moet snappen in welke collectie gezocht moet worden voor welk type relatie.
  + **Moeilijk voor generieke traversals**
    - “Geef alle Nodes die 3 stappen verwijderd zijn van node X”, dan zal je zelf over meerdere collecties heen moeten traversen.
  + **Minder flexibel bij nieuwe relatie-types**
    - Voor elke relatie moet je een **nieuwe collectie** aanmaken, wat kan leiden tot veel collecties als het model groeit.

**Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, menu, Lettertype

Automatisch gegenereerde beschrijving**

**4. Nodes in verschillende collecties en relaties in 1 collectie**

* Voordelen
  + **Duidelijke scheiding:**
    - Nodes en relaties apart houdt het model strak en vergelijkbaar met een echte graph DB.
  + **Geen duplicatie:**
    - Je hoeft realites niet in beide Nodes bij te houden, want je hebt een aparte tabel/collectie.
  + **Relaties kunnen properties hebben:**
    - Handig voor bijvoorbeeld gewicht van een joint, timestamp, sterkte van connectie.
  + **Flexibel:** 
    - Makkelijk nieuwe soorten relaties toevoegen zonder Nodes te veranderen.
  + **Schaalbaarder:**
    - Dan arrays in Nodes, omdat relaties onafhankelijk zijn.
* Nadelen
  + **Meer queries nodig**
    - Om een node met zijn relaties te halen, moet je vaak eerst de node ophalen, daarna alle edges, daarna de target Nodes.
  + **Geen native joins in document-databases**
    - Dus je moet dit in je applicatie-logica doen (of via aggregation pipelines, wat soms ingewikkeld is).
  + **Performance**
    - Bij veel hops kan dit zwaar worden, omdat je elke stap zelf moet oplossen in code of aggregaties.
  + **Complexer te bouwen API**
    - Je moet endpoints maken die Nodes in relaties apart afhandelenAfbeelding met tekst, schermopname

      Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype

      Automatisch gegenereerde beschrijving