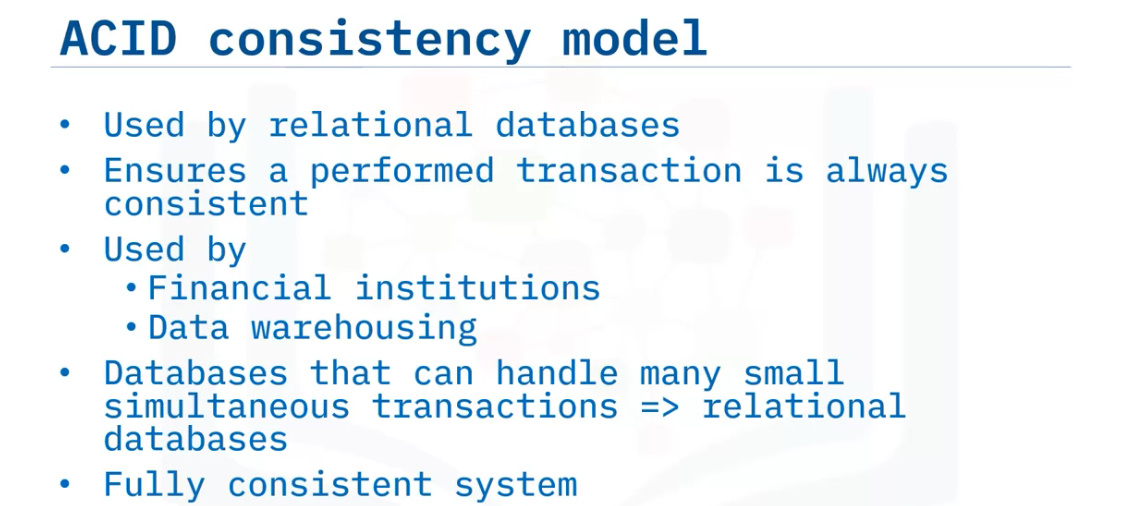
**Consistency Models**

* Relation DB – ACID



Sunt foarte mult folosite in Institutii bancare

* NoSQL – BASE
* **NoSQL se refera mai mult la baza de date distributive**

**BASE**

* **Basically Available** – garanteaza disponibilitatea datelor. Orice request va avea mereu un response, chiar si de fail. NoSQL se bazeaza ca datele sa fie mereu disponibile, si foloseste un approach foarte **distributiv** pentru storage, adica in loc de a pastra toate datele pe un singur system storage, datele sunt stocate pe mai multe medii de stocare. Daca gen niste date nu pot fi accesate de pe un storage, nu cade toata baza de date.
* **Soft State** – datele nu mai sunt consistente si pot aparea orice anomalii. Tranzactiile se executa fara vreun lock.
* **Eventually consitency** – sistemul devine consistent odata ce nu mai primeste input. Deci, asta nu inseamna ca NoSQL nu e niciodata consistent.

**BASE Consistency model**

* NoSQL are putine cerinte pentru immediate consistency, date proaspete sau acuratete
* BASE se bazeaza mai mult pe aceea ca datele trebuie sa fie mereu disponibile, macar si o parte din ele
* In schimb, ofera beneficii ca:

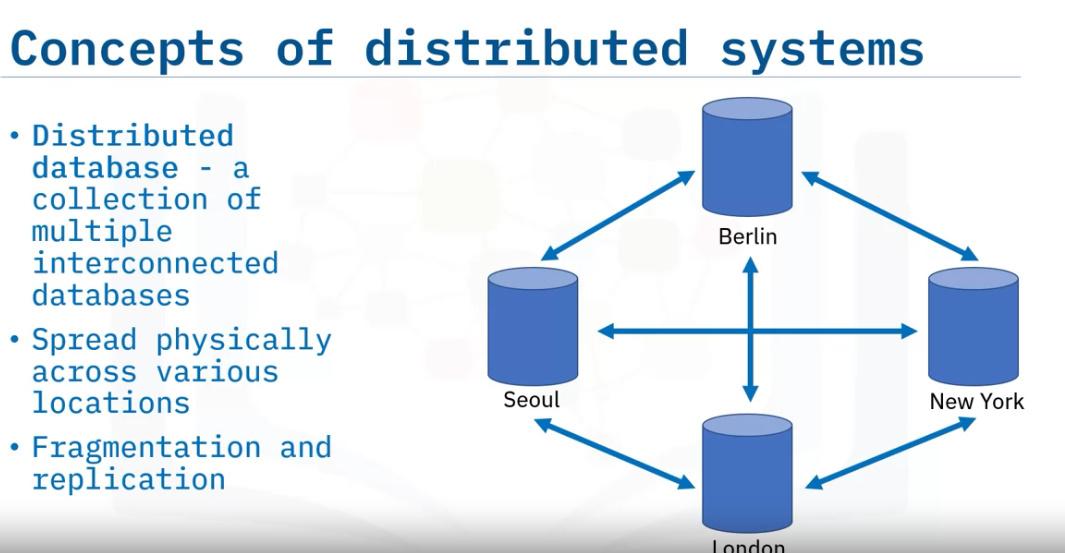
- availability

- scale

- resilience

* Sunt mai degraba pentru disponibilitate decat consistenta
* Deci, datele nu sunt mereu proaspete, consistente sau cele mai actuale
* Sunt des utilizate in Social media, unde se stocheaza date imense, dar ce nu trebuie sa fie neaparat consistente

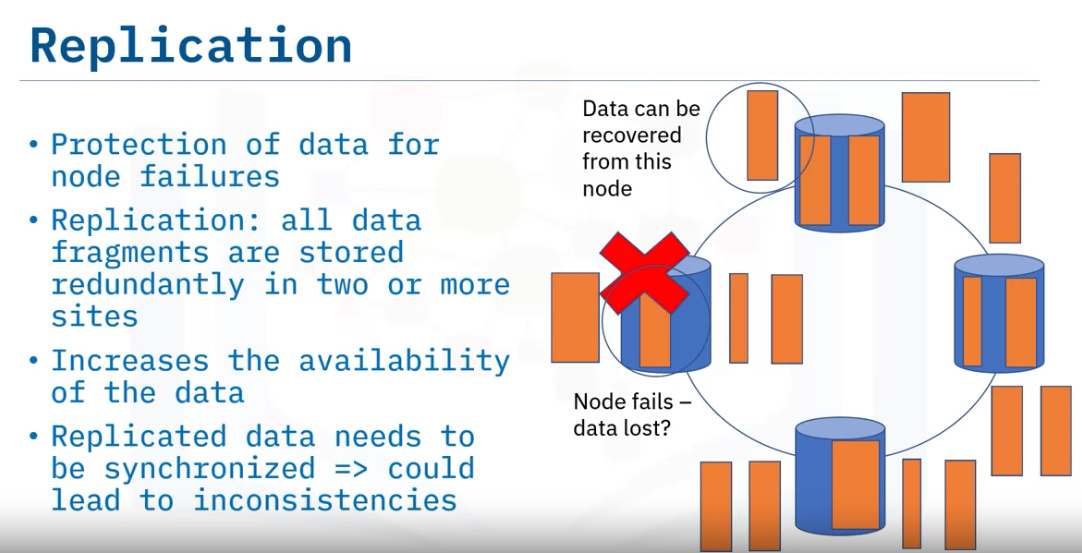
**Distributed Databases**

* **Distributed Database** – colectie de baze de date interconectate separate fizic in diferite locatii si comunica prin computer network, asa incat pare ca e o singura baza de date
* Fiecare locatie unde o baza de date din colectie ruleaza se numeste **node**
* ****
* O baza de date distribuita este fizic distribuita prin **fragmentation** sau **replication** the data
* **Replication** – toate datele sunt stocate pe mai multe servere disponibile in mod redutant. Adica, mai multe servere, sau toate, vor contine aceleasi date, deci copii identice ale datelor. Ca si cum, avem aceeasi baza de date, dar pe diferite servere, si asta e redutant.

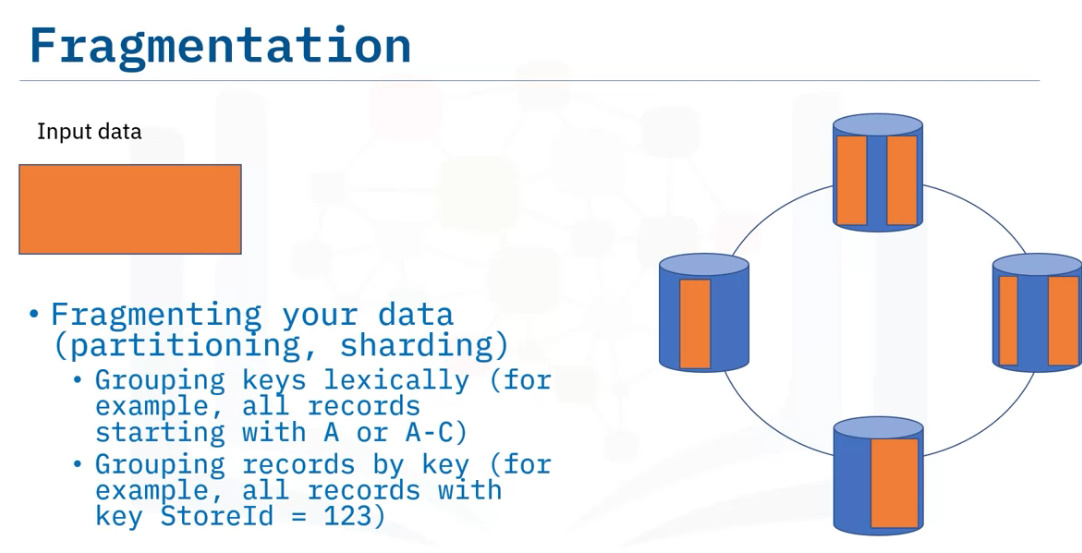
- daca cade un server, celelalte pot inca oferi toate datele

- pot fi rulate queries in mod paralel.

- problema e ca datele trebuie merreu actualizate pe toate serverele, adica daca un server niste date se modifica, ele trebuie modificate pe toate, in caz contrar apare inconsistenta de date.



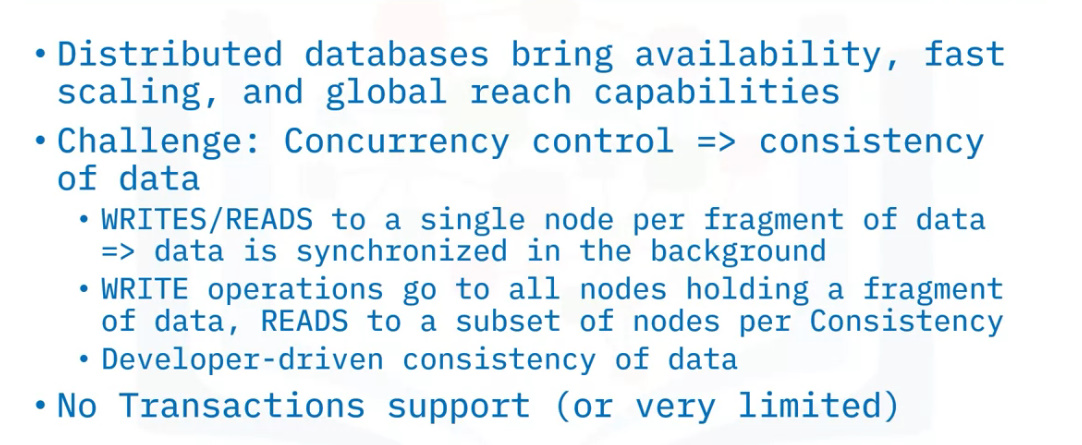
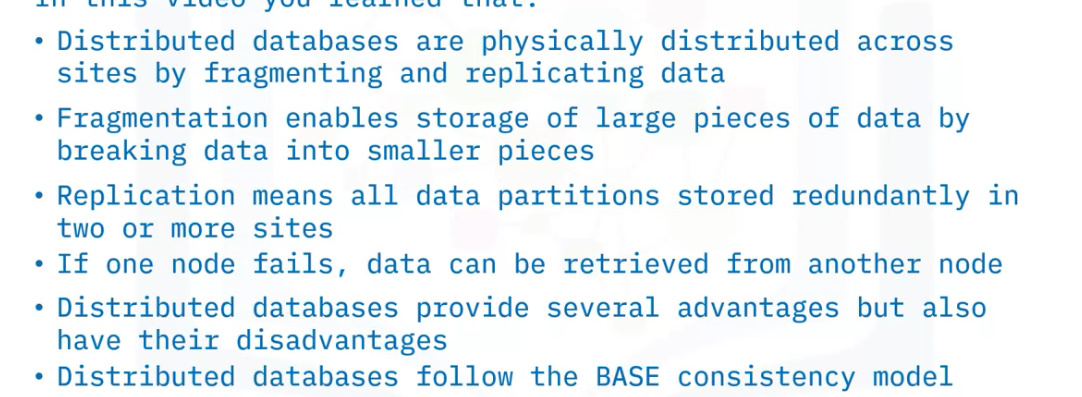
* **Fragmentation** – datele sunt impartite in fragmente, deci fiecare server stocheaza bucati de date, nu copii repetate.

****

Datele pot fi fragmentate de ex, dupa literele alfabetului, gen primul server ia datele ce au id ce incepe cu 1, al doilea server datele ce au id ce incepe cu 2(sau key in loc de id)

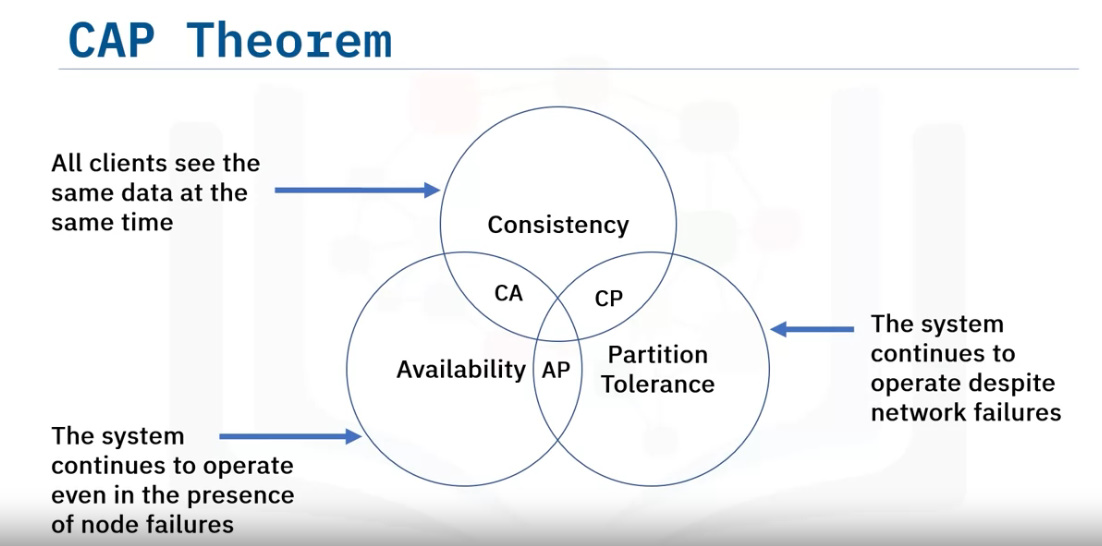
- Ideea e ca cu fragmentation, din bucatile de date de pe diferite servere, adunadu-le pe toate, refacem baza de date

- Insa, problema e ca data un server cade, poate sa cada toata baza de date

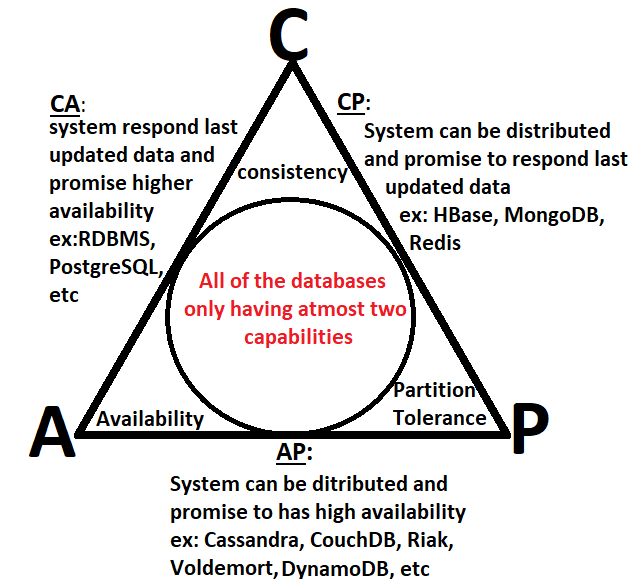
* Distributed DB urmeaza BASE model
* 
* 

**CAP Theoreme**

* **Se refera la sisteme distribuite cu replicare**
* **Consistency** – se refera la aceea ca fiecare node are exact aceleasi date, cele mai recente, deci oricare node nu ar fi folosit, va returna date identice ca altele. Cu alte cuvinte, toti clientii vad aceleasi date
* **Availability** – orice request mereu va primi un raspuns, chiar daca unele noduri sunt down, caci va primi datele de la alte noduri.
* **Partition tolerance** – chiar daca apar anumite rupturi de conexiune (**network)** intre 2 sau mai multe noduri, sistemul oricum va continua sa lucreze asa cum este.Deci, nu conteaza ce network issues apar, oricum sistemul va functiona.**Partition** inseamna o pierdere de conexiune intre noduri.
* Intr-un distributed system e imposibil de a evita partitions!Partition tolerance este o caracteristica esentiala a NoSQL

****

* **CAP Theoreme** prevede ca un sistem distribuit poate avea ***doar 2 din cele 3*** caracteristici CAP!!!
* Atentie! Termenul de Consistency in CAP inseamna cu totul altceva fata de ce inseamna el in ACID. In CAP, consistency inseamna ca toate nodurile au date identice, in ACID inseamna ca o tranzactie nu va viola constrangerile de integritate,



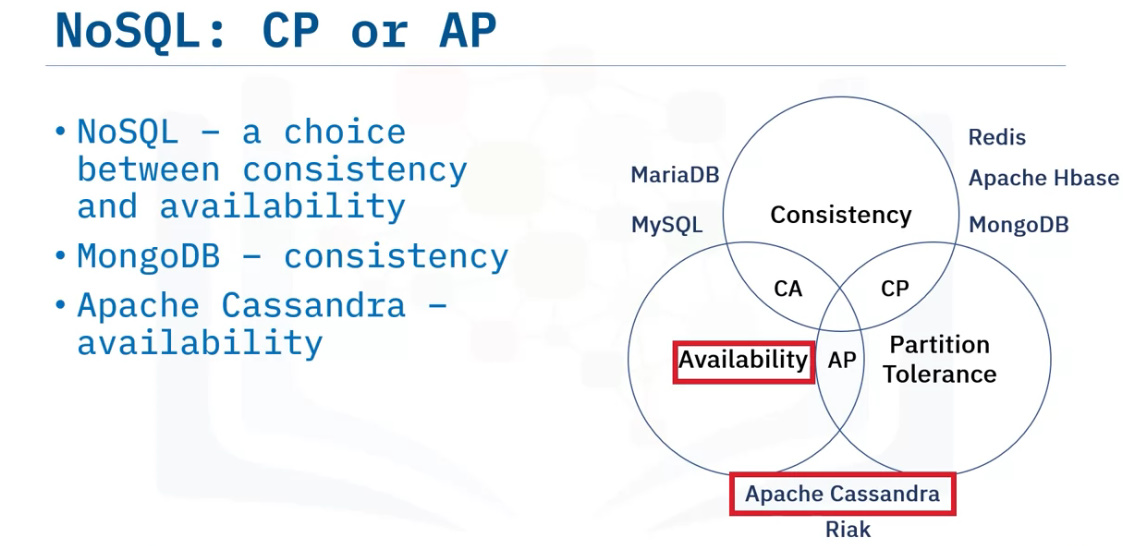
- CP – sistemul ofera mereu date consistente, indiferent de node si indiferent ca intre anumite noduri s-a pierdut conexiunea. Daca vreun node cade, e problema.

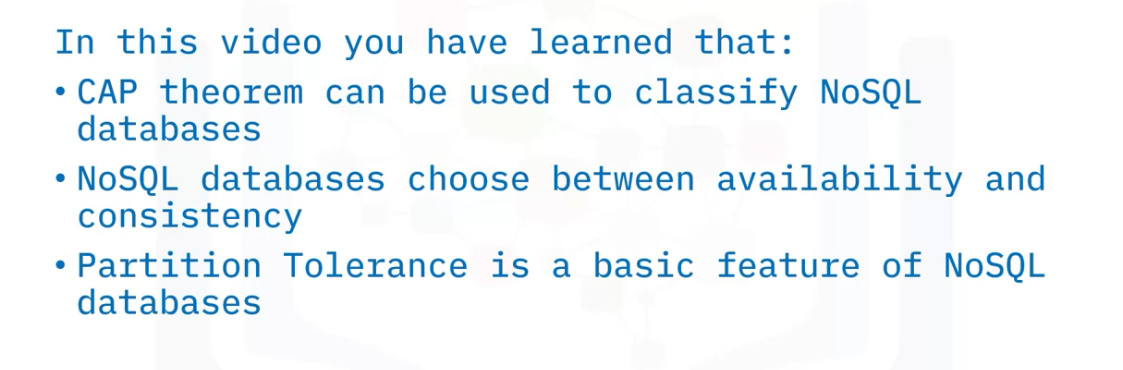
- AP – Sistemul ofera date mereu, chiar daca un nod cade, caci el e scos din functiune, si datele se returneaza cat mai repede, dar deja nu se mai tine cont de consistency.

- CA – datele sunt consistente si rapid returnate, dar daca se pierde legatura network intre 2 sau mai multe noduri, sistemul poate da fail

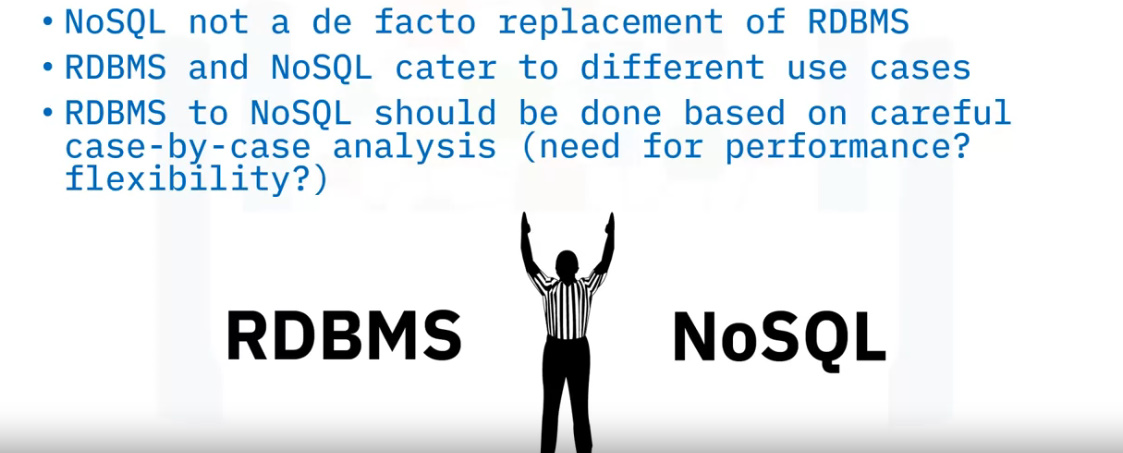
* Deci un sistem distributiv nu poate oferi si consistenta si availability totala in acelasi timp!

**NoSQL CP or AP**

* Pentru NoSQL Partition Tolerance este primordial, deci nu mai putem alege CA
* 
* Asta nu inseamna ca MongoDB nu ofera Availability, de ex, doar ca nu se focuseaza asa mult pe ea.

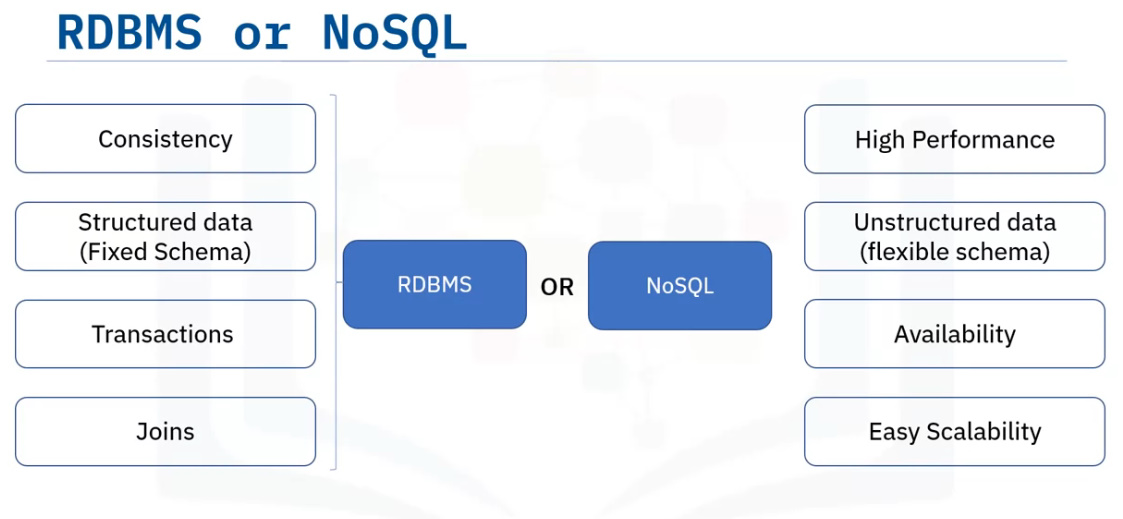


**Probleme la migrarea de la RDBMS la NoSQL**

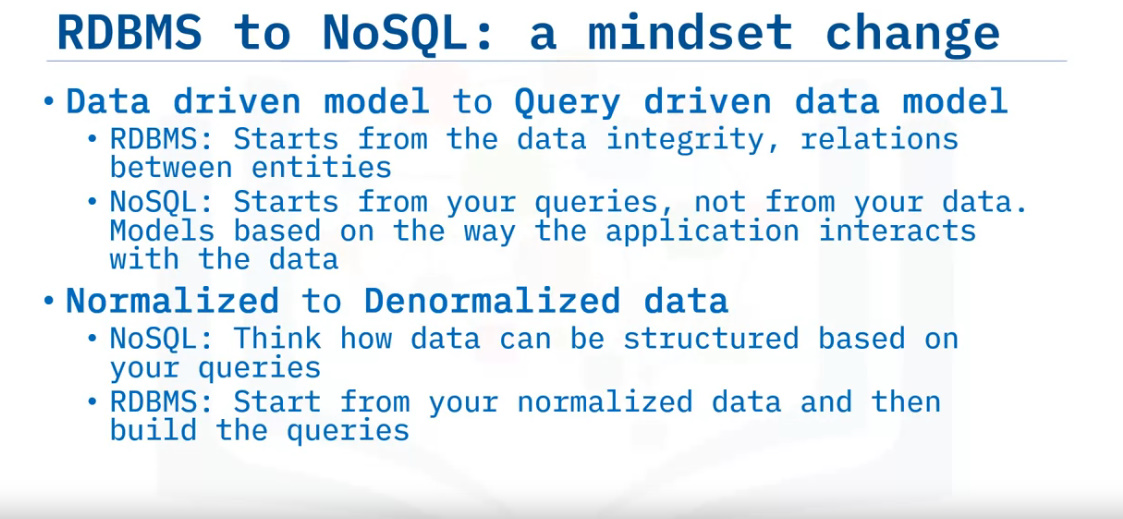


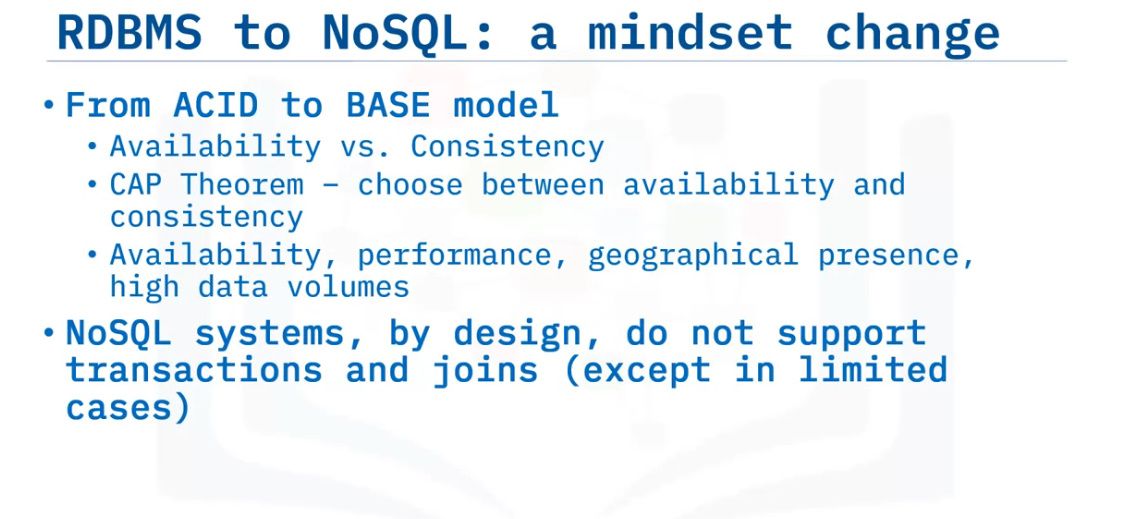
NoSQL nu este un competitor pentru RDMBS. El doar se foloseste in diferite situatii fata de RDBMS

* Alegerea lor depinde strict de cerintele noastre



* Pot fi si sisteme ce folosesc si RDBMS si NoSQL





* Mai multe sisteme acum se bazeaza pe availability
* NoSQL nu prea suporta joins!

