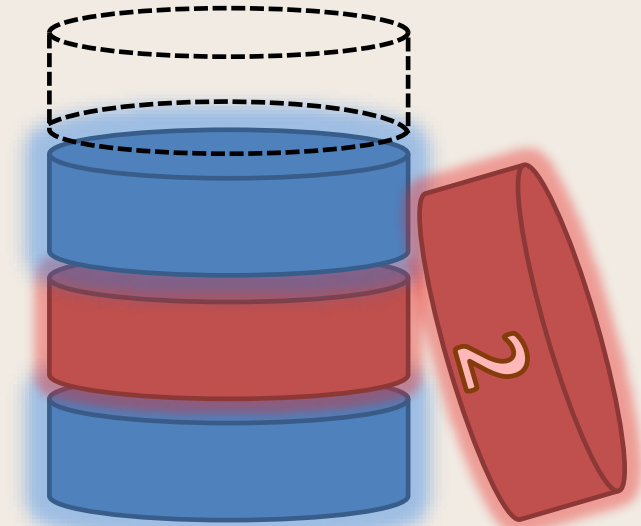


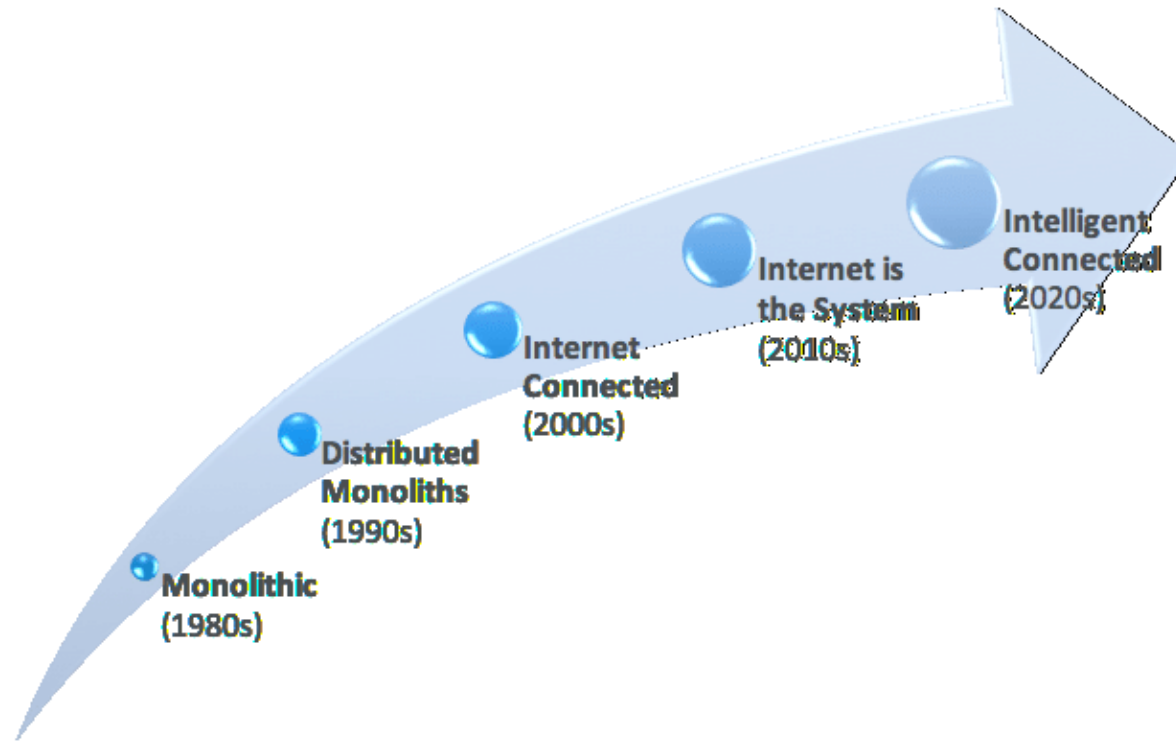
# Baze de date NoSQL





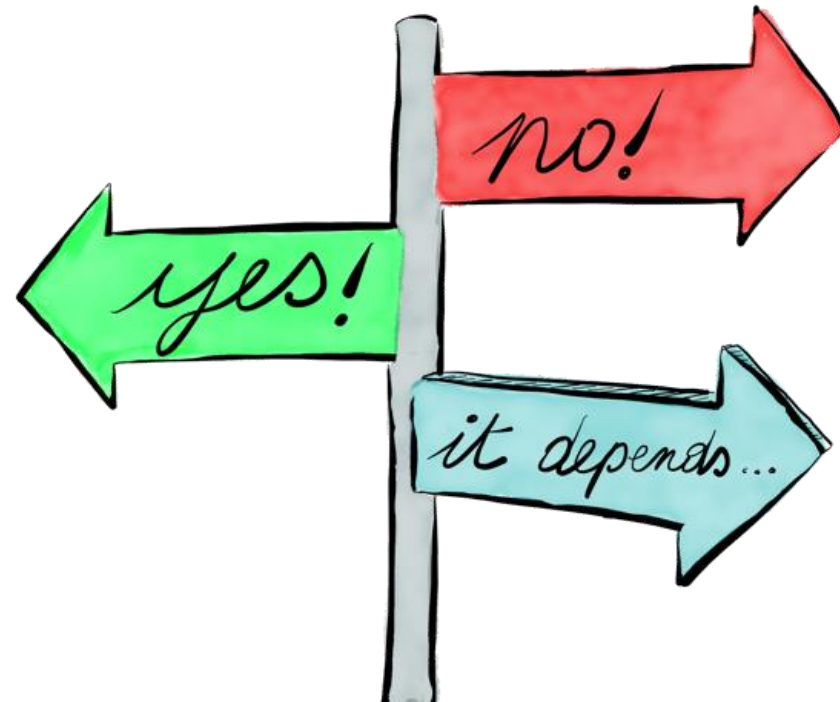
# Context

- Modelul relațional are aproape 50 de ani
- ACID – asigură robustețea procesărilor tranzacționale dar cu costuri de performanță



# Context

- Trăsăturile cele mai căutate pentru o bază de date:
  - Scalabilitate (verticală/orizontală)
  - Disponibilitate (*"five nines" availability*)
  - Performanță



# Ce înseamnă NoSQL?

- Orice bază de date ce nu folosește SQL
  - totuși, nu include bazele de date orientate-obiect
- *Not Only SQL*
  - Cassandra: limbaj de interogare *java-like*, CQL
- O bază de date ce:
  - NU folosește conceptele modelului relațional pentru stocarea datelor
  - NU permite accesarea datelor prin intermediul limbajului SQL standard

# Ce înseamnă NoSQL?

- Exemple de baze de date NoSQL (peste 225):

<https://hostingdata.co.uk/nosql-database/>



“Următoarea generație de baze de date acoperă, în general, următoarele aspecte: modelare non-relațională, distribuit, *open-source*, scalabilitate orizontală”

# Diferențe majore de abordare

- BD Relaționale:

- informațiile sunt extrase folosind operații de join,
- accelerarea procesării presupune deseori indexare,
- proprietățile ACID sunt impuse

- Alternativă: Teorema CAP

- **Consistency** (nu dpdv al respectării constrângerilor de integritate, ci din punct de vedere al al furnizării acelorași date tuturor clienților)
- **Availability** – nivelul de disponibilitate crește odată cu creșterea numărului de noduri redundante
- **Partition Tolerance** – găsirea de rute alternative în rețea pentru a obține date din diverse noduri

# Modelul de consistență BASE

- **B**asic **A**vailability

- Baza de date (pare că) funcționează în marea majoritate a timpului

- **S**oft-state

- Consistența la scriere nu e necesară. Replicile nu trebuie să fie mutual consistente

- **E**ventual consistent

- Baza de date va fi consistentă la un moment dat

*Read Repair – Delayed Repair*




# Abordări bazate pe familii de coloane

- BD Relaționale sunt abordări bazate pe linii
- Foarte frecvent însă aplicațiile ce accesează o bază de date interoghează date memorate pe o coloană
- În ciuda optimizărilor memorarea într-o zonă continuă a valorilor aceleași coloane e mai performantă

# Abordări bazate pe coloane

1. find the start of the row



9645	9645	STANSTED	FRANCE	PERPIGNAN	RYANAIR	A	S	7.73885350318471
9646	9646	STANSTED	FRANCE	PERPIGNAN	RYANAIR	D	S	4.53548387096774
9647	9647	STANSTED	FRANCE	POITIERS	RYANAIR	A	S	5.0981308411215
9648	9648	STANSTED	FRANCE	POITIERS	RYANAIR	D	S	5.83255813953488
9649	9649	STANSTED	FRANCE	RODEZ	RYANAIR	A	S	8.96045197740113
9650	9650	STANSTED	FRANCE	RODEZ	RYANAIR	D	S	4.41242937853107
9651	9651	STANSTED	FRANCE	TARBES-LOURDES INTERNATIONAL	AIR MEDITERRANEE	A	C	67.5
9652	9652	STANSTED	FRANCE	TARBES-LOURDES INTERNATIONAL	AIR MEDITERRANEE	D	C	67
9653	9653	STANSTED	FRANCE	TARBES-LOURDES INTERNATIONAL	EASTERN AIRWAYS	A	C	49
9654	9654	STANSTED	FRANCE	TARBES-LOURDES INTERNATIONAL	JET2.COM LTD	A	C	0
9655	9655	STANSTED	FRANCE	TARBES-LOURDES INTERNATIONAL	JET2.COM LTD	D	C	0
9656	9656	STANSTED	FRANCE	TARBES-LOURDES INTERNATIONAL	RYANAIR	A	S	8.375
9657	9657	STANSTED	FRANCE	TARBES-LOURDES INTERNATIONAL	RYANAIR	D	S	4.70833333333333
9658	9658	STANSTED	FRANCE	TARBES-LOURDES INTERNATIONAL	TITAN AIRWAYS LTD	A	C	18.1470588235294
9659	9659	STANSTED	FRANCE	TARBES-LOURDES INTERNATIONAL	TITAN AIRWAYS LTD	D	C	14.4444444444444


The rdbrms approach

2. then move to the column  
and get the value



3. repeat for every row

9645, 9646, 9647, 9648				
etc				
PERPIGNAN , PERPIGNAN, POITIERS, POITIERS				
etc				
7.73885350318471 , 4.5354838709677, 5.0981308411215, 5.83255813953488				



Just read the column data sequentially

# Cassandra



## ■ Column Family

- = colecție de coloane ce sunt accesate împreună de cele mai multe ori
- Corespondentul tabelului din modelul relațional
- Stocat într-un fișier distinct și sortat după valoarea cheii

## ■ Column

- Unitatea de stocare
- Are un nume unic, o valoare și un *timestamp*

## ■ Timestamp

- Pentru rezolvarea de conflicte. E furnizat de client
- Reprezintă numărul de milisecunde scurs de la 1 Ianuarie 1970

## ■ SuperColumn

- Listă de coloane (asemănător unui *view*)

# Exemplu de Column Family



- Colecție de linii aeriene din Marea Britanie.
- Coloane: *Airline Name, Km Flown (x1000), No of Flights, No of Hours flown, Number of Passengers handled*

The screenshot shows the OpenOffice.org Calc application window titled 'domesticflightsJan.csv - OpenOffice.org Calc'. The spreadsheet displays data for 12 airlines. The columns are labeled A, B, C, D, and an unlabeled column for passengers. The data is as follows:

	A	B	C	D	
1	AURIGNY AIR SERVICES	▶ 193	1388	887	26585
2	BA CITYFLYER LTD	▶ 300	545	686.1	30031
3	BLUE ISLANDS LIMITED	▶ 168	991	520.8	15308
4	BMI GROUP	▶ 1067	2435	2922.7	142804
5	BRITISH AIRWAYS PLC	▶ 1510	3327	4116.6	307849
6	BRITISH INTERNATIONAL HELICOPTER SERVICES LTD	▶ 10	162	57.9	2169
7	EASTERN AIRWAYS	▶ 496	1406	1353	23074
8	EASYJET AIRLINE COMPANY LTD	▶ 1826	3922	4297.2	399308
9	FLYBE LTD	▶ 2505	6755	5635.4	297435
10	ISLES OF SCILLY SKYBUS	▶ 12	176	55.3	1200
11	JET2.COM LTD	▶ 22	71	65	4059
12	LOGANAIR	▶ 504	2440	1958.7	32994

# Crearea unui *Column Family*



```
Create Column Family DomesticFlights
WITH comparator = UTF8Type AND
key_validation_class = UTF8Type AND
column_metadata =
[
    {column_name: airline, validation_class: UTF8Type, index_type: KEYS},
    {column_name: Kms, validation_class: IntegerType},
    {column_name: Flights, validation_class: IntegerType},
    {column_name: Hrs, validation_class: FloatType},
    {column_name: Pass, validation_class: IntegerType}
];
```

# Inserare date



```
set DomesticFlights['Aurigny Air Services']['Kms'] = 193; set  
DomesticFlights['Aurigny Air Services']['Flights'] = 1388; set  
DomesticFlights['Aurigny Air Services']['Hrs'] = 887; set  
DomesticFlights['Aurigny Air Services']['Pass'] = 26585;
```

```
set DomesticFlights['BA CityFlyer']['Kms'] = 300;  
set DomesticFlights['BA CityFlyer']['Flights'] = 545;  
set DomesticFlights['BA CityFlyer']['Hrs'] = 686;  
set DomesticFlights['BA CityFlyer']['Pass'] = 30031;
```



# Regăsirea datelor



## ■ LIST DomesticFlights

```
-----  
RowKey: BA CityFlyer
```

```
=> (column=Flights, value=545, timestamp=1354875194019000)
```

```
=> (column=Hrs, value=686.0, timestamp=1354875194033000)
```

```
=> (column=Kms, value=300, timestamp=1354875194010000)
```

```
=> (column=Pass, value=30031, timestamp=1354875201897000)
```

```
-----  
RowKey: Aurigny Air Services
```

```
=> (column=Flights, value=1388, timestamp=1354875193983000)
```

```
=> (column=Hrs, value=887.0, timestamp=1354875193991000)
```

```
=> (column=Kms, value=193, timestamp=1354875193958000)
```

```
=> (column=Pass, value=26585, timestamp=1354875194001000)
```

# Regăsirea datelor



## ■ GET DomesticFlights['BA CityFlyer']

```
=> (column=Flights, value=545, timestamp=1354875194019000)
=> (column=Hrs, value=686.0, timestamp=1354875194033000)
=> (column=Kms, value=300, timestamp=1354875194010000)
=> (column=Pass, value=30031, timestamp=1354875201897000)
```



# Modificare *Column Family*



```
UPDATE COLUMN FAMILY DomesticFlights
WITH comparator = UTF8Type AND
key_validation_class = UTF8Type AND
column_metadata =
[
    {column_name: airline, validation_class: UTF8Type, index_type: KEYS},
    {column_name: Kms, validation_class: IntegerType, index_type: KEYS},
    {column_name: Flights, validation_class: IntegerType},
    {column_name: Hrs, validation_class: FloatType},
    {column_name: Pass, validation_class: IntegerType}
];
```



## Ștergere date

```
del DomesticFlights['BA CityFlyer']['Hrs'];
```

```
del DomesticFlights['BA CityFlyer'];
```

## Ștergere Column Family

```
drop column family DomesticFlights;
```

# CQL Crearea unui *Column Family*



```
CREATE KEYSPACE Flights WITH strategy_class = SimpleStrategy
AND strategy_options:replication_factor = 1;

use Flights;

create ColumnFamily FlightDetails
(airline varchar PRIMARY KEY,
Kms int,
Noflights int,
Hrs float,
Pass int);

copy FlightDetails (airline, Kms, Noflights, Hrs, Pass) from 'domDataOnly.csv';

select * from FlightDetails;
```

# CQL Regăsirea datelor



```
use Flights;  
select count(*) from Airports where CountryCode = 'GB' and Lat > 51;
```

# Abordări bazate pe documente

- nu există un design al bazei de date în adevăratul sens al cuvântului
- bazele de date nu au o structură și nici constrângeri de integritate. (nici măcar de tip)
- *sharding* - partiționarea unei baze de date foarte mari în părți de dimensiuni reduse și care sunt mai ușor și mai rapid de gestionat. Fiecare *shard* e memorat pe un nod ce are propria sa instanță activă de bază de date
- MongoDB, CouchDB

# MongoDB

- *Database* – colecție de date înrudite
- *Collection* – container pentru documente
- *Document* – o componentă a colecției
- *Field* – similar cu modelul relațional
- *Embedded document* – cel mai potrivit corespondent din modelul relațional este *join*-ul
- Primary key
- Secondary key

# Adăugare date

```
> use Airlines
switched to db Airlines
> Airline12 = {"Name" : "LOGANAIR ", "Km": 504 , "NoFlights" : 2440, "Hrs" : 1958.7 , "NoPass" : 32994 }
{
  "Name" : "LOGANAIR ",
  "Km" : 504,
  "NoFlights" : 2440,
  "Hrs" : 1958.7,
  "NoPass" : 32994
}
> db.Flights.insert( Airline12 )
> db.Flights.find()
{ "_id" : ObjectId("50cb3e02066f55d5e394ec1a"), "Name" : "LOGANAIR ", "Km" : 504, "NoFlights" : 2440, "Hrs" : 1958.7, "NoPass" : 32994 }
>
```

Regăsire date

```
db.Flights.find( { "Km": 504 })
```



## Regăşire date

```
Airline7 = { "Name": "EASTERN AIRWAYS", "Km": 496, "NoFlights":  
1406, "Hrs": 1353, "NoPass": 23074 }  
db.Flights.insert( Airline7 )
```

```
Airline8 = { "Name": "EASYJET AIRLINE COMPANY L", "Km": 1826,  
"NoFlights": 3922, "Hrs": 4297.2, "NoPass": 399308 }  
db.Flights.insert( Airline8 )
```

```
Airline9 = { "Name": "FLYBE LTD", "Km": 2505, "NoFlights": 6755,  
"Hrs": 5635.4, "NoPass": 297435 }  
db.Flights.insert( Airline9 )
```

```
Airline10 = { "Name": "ISLES OF SCILLY SKYBUS", "Km": 12, "NoFlights":  
176, "Hrs": 55.3, "NoPass": 1200 }  
db.Flights.insert( Airline10 )
```

```
Airline11 = { "Name": "JET2.COM LTD", "Km": 22, "NoFlights": 71,  
"Hrs": 65, "NumPass": 4059 }  
db.Flights.insert( Airline11 )
```

## Regăsire date

```
db.Flights.remove({NumPass:4059})
```

Regăsire date

```
db.getCollectionNames();
```

## Regăsiți date

```
db.Flights.find({ $and: [ { Km: { $gt: 2000 } }, { NoPass: { $gt: 140000 } } ] } )
```

```
db.Flights.find({ $or: [ { Km: { $gt: 2000 } }, { NoPass: { $gt: 140000 } } ] } )
```

```
db.Flights.find( { Km: { $in: [ 300, 496 ] } } )
```

```
db.Flights.find().sort({ Km: -1 })
```

```
db.Flights.find().sort({ Km: -1 }).limit(1)
```

```
db.Flights.find( { Animals: { $exists: true } } )
```

# Indexare

- structura indecșilor MongoDB este de B-arbore

```
db.Flights.ensureIndex( { Km: 1 } )
```

```
db.Flights.find().hint( { Km: 1 } )
```

```
db.Flights.getIndexes()
```

# Actualizarea datelor

```
db.Flights.update( { Km: 11 }, { $set: { Animals: "Elephants and Badgers" } })
```

```
db.Flights.update( { Km: 112 }, { $rename: { Animals: "Creatures" } })
```

```
db.Flights.update ( { Km: 112 }, { $set: { Rivers: "Don and Ouse" } })
```