to SQL, or Not to SQL

that is the question

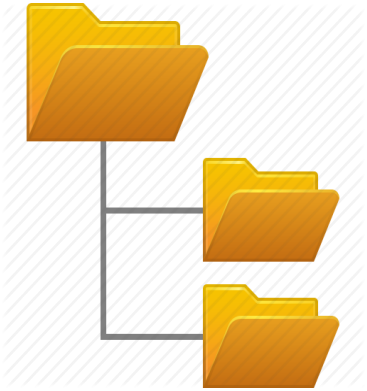
Based on slides origially created by
Sandesh Sanjay Gade (Previous TA)

UNIVERSITY OF VIRGINIA
DATA SCIENCE
INSTITUTE

What Do You

THINK???

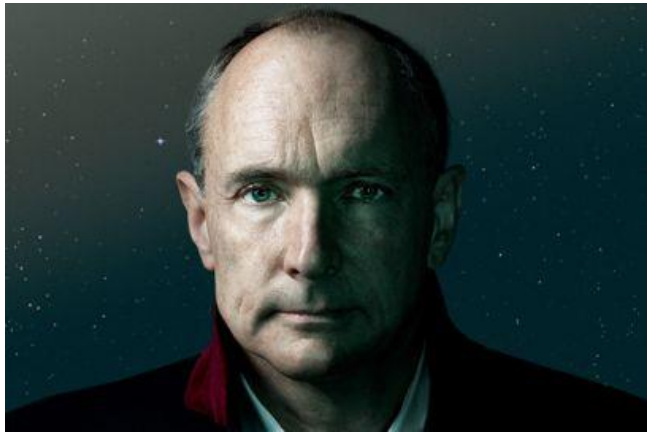
How would **YOU** represent unstructured data?



File System… anyone?

What is the largest publicly accessible **distributed** collection of **unstructured documents** in the world?

Hint:
Sir Tim Berners Lee



The

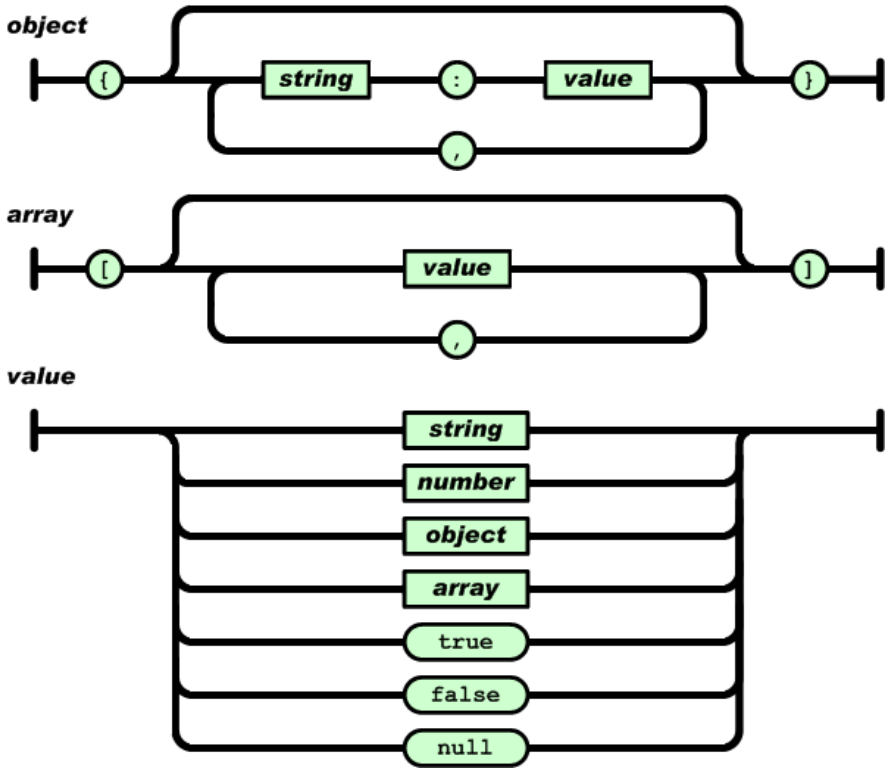
Why NoSQL?

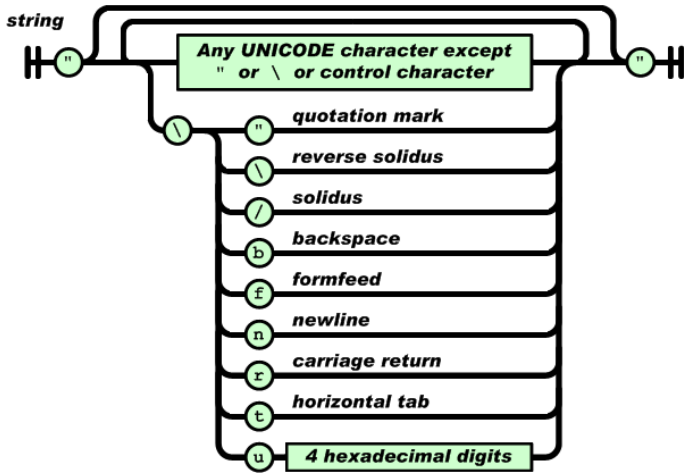

DO YOU HAVE ANY EXPERTISE IN SQL?
NO
geek & poke
DOESN'T MATTER. WRITE: "EXPERT IN NO SQL"
Leverage the NoSQL boom

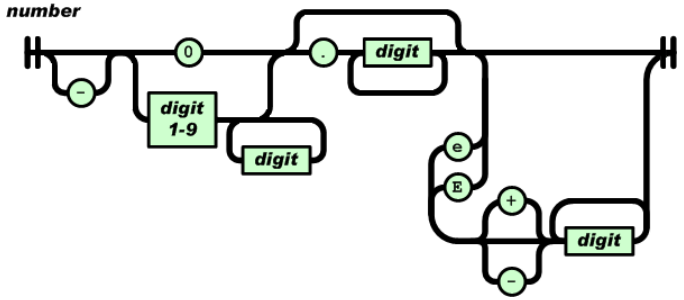
Why NoSQL?

- Not all data can be expressed in relational manner.
  - RDBMS can model data **ONLY** in the form of tabular relations.
  - Data can be expressed in other structures such as ***key-value***, ***wide column***, ***graph*** or ***document***.
  - How would **YOU** express data that does not naturally conform to a tabular representation?
    - CAN YOU? **;)**
    - WOULD YOU? **o.0**
    - SHOULD YOU? **-_-**
- Simplicity of design
- Scalability
  - Concept of “horizontal” scaling to clusters of machines.
    - ^ BIG problem for relational databases.

JSON (JavaScript Object Notation)







How JSON sparked NoSQL

- Features of JSON:
  - *Data structure of the **web***.
  - **Simple** data format.
  - Displaced more complex formats such as XML and *ML.
  - **Developer Friendly** (Supported in virtually every programming language)
  - **Agility** of JSON records.
  - **Lack of predefined schema** makes upgrades easy.


```
"firstName": "John",
"lastName": "Smith",
"age": 25,
"address": {
    "streetAddress": "21 2nd S
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
},
"phoneNumbers": [
    {
        "type": "home",
```

SQL vs NoSQL
DIFFERENCES

SQL Databases are primarily called **Relational Databases (RDBMS)**
Table based databases.

Have **predefined schema**.
Vertically Scalable (***#ScaleUP***)
SQL DB's use SQL, duh. SQL is very powerful defining and manipulating data.

NoSQL databases are primarily **non-relational** or **distributed databases**.
Document, **key-value pair**, **graph** or **wide-column** based databases.
Dynamic Schema for unstructured data.
Horizontally Scalable (***#ScaleOUT***)
In NoSQL databases, queries are focussed on a collection of *{insert term here}*

SQL vs NoSQL
DIFFERENCES

SQL databases are a good fit for a **complex query intensive environment.**

SQL databases are **not best fit for hierarchical data** storage.

SQL databases are **best fit for heavy duty transactional type applications**, as it is more stable & promises atomicity, integrity of the data.

NoSQL databases don't have standard interfaces to perform complex queries, hence the queries themselves in NoSQL are not as powerful.

NoSQL database **fits better for the hierarchical data storage** requirement as it follows the key-value pair way of storing data (***#JSONftw***)

Although, NoSQL **can be tuned for transactions**, it is not comparable in terms of stability (on high load) and for complex transactional applications.

SQL vs NoSQL

Consistency…?

Emphasis on **ACID** (Atomicity, Consistency, Isolation and Durability) Properties

Relational **SQL** databases are very **strongly consistent** (“C” in ACID)

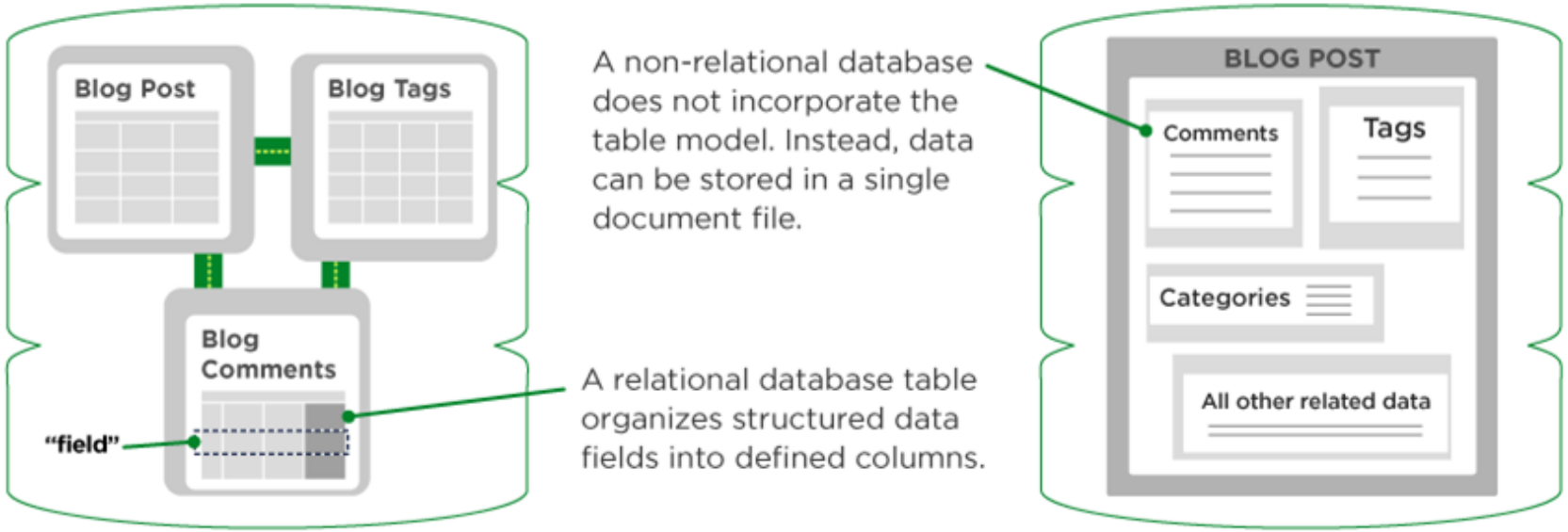
Emphasis on Brewers **CAP** (Consistency, Availability and Partitions tolerance) theorem.

In **NoSQL** the **consistency varies** depending on the type of DB. For example

In GraphDB such as Neo4J, consistency ensures that a relationship must have a start and end node

In MongoDB, it automatically creates a unique rowid, using a 24bit length value

Blogging website example: Relational vs Non-relational DB ("SQL" vs "NoSQL")



SQL vs NoSQL
PROS

Relational databases are good at **structured data** and **transactional, high-performance workloads.**

Offerings are proven and mature with a wide variety of tools available

NoSQL databases are good for non-relational data. **Schema-less architecture** allows for frequent changes to the database and easy addition of varied data to the system.
Allows for *heterogeneous* items (maybe not all blog posts have associated photos or video, etc…) This difference is okay! Very flexible!

Easily scalable (*horizontally*), runs well on distributed systems (***#CloudFirst***)

NoSQL [large document object / “file”] PROS

Large objects, or “files” are easily stored in MongoDB. It is no problem to store 100MB videos in the database.

This has a number of advantages over files stored in a file system. Unlike a file system, the database will have **no problem dealing with millions of objects.** Additionally, we get the power of the database when dealing with this data: we **can do advanced queries** to find a file, using indexes; we can also do neat things like **replication** of the entire file set.

[https://www.mongodb.com/blog]

SQL vs NoSQL
CONS

Can be difficult to scale.

Fixed schema for organizing data.

Installation, management and toolsets are still maturing

Can have slower response time.

SQL vs NoSQL
EXAMPLES

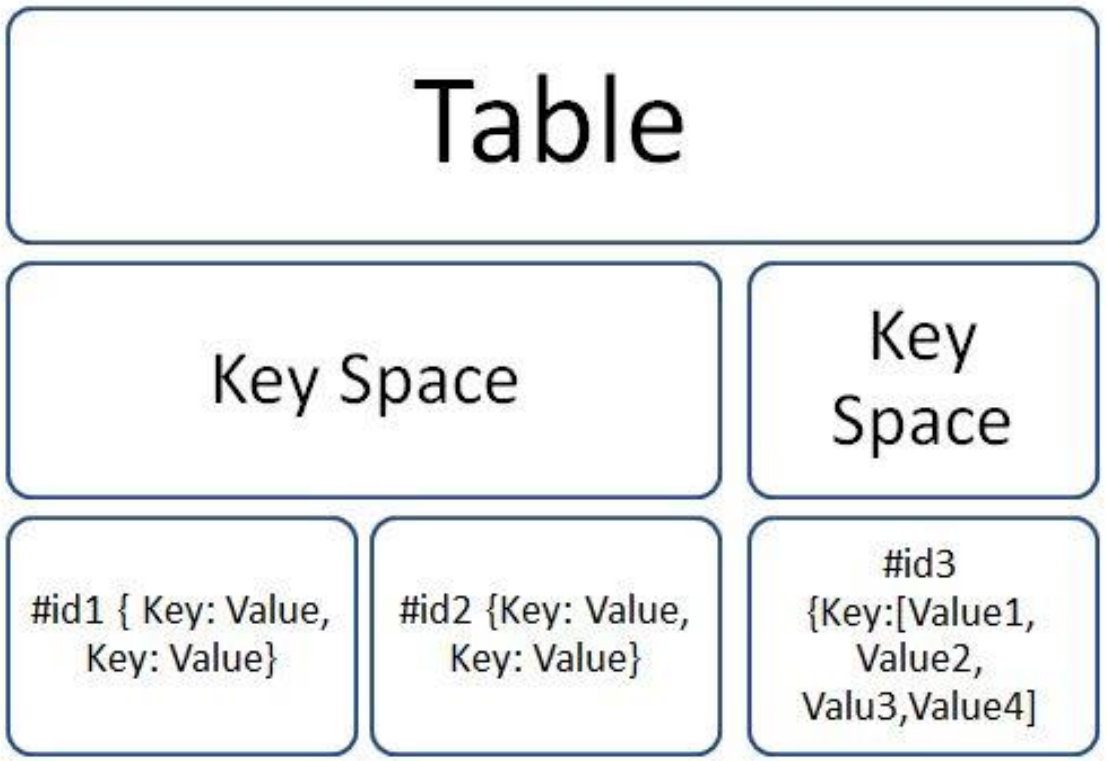
MySQL
PostGreSQL
SQL Server
Oracle
SQLite

MongoDB
Amazon DynamoDB
Couchbase
Neo4j
Redis

Types of NoSQL Databases

Key-Value Store

- Contains many *key spaces*.
- Each ***key space*** can have multiple ***identifiers*** to store ***key-value pairs***
- Suitable for building simple, non-complex, high available applications.
- *E.g.* ***Amazon DynamoDB, Redis, Riak***



```
> set mykey somevalue
OK
> get mykey
"somevalue"
```

```
> mset a 10 b 20 c 30
OK
> mget a b c
1) "10"
2) "20"
3) "30"
```

```
> set counter 100
OK
> incr counter
(integer) 101
> incr counter
(integer) 102
> incrby counter 50
(integer) 152
```

```
> set key some-value
OK
> expire key 5
(integer) 1
> get key (immediately)
"some-value"
> get key (after some time)
(nil)
```

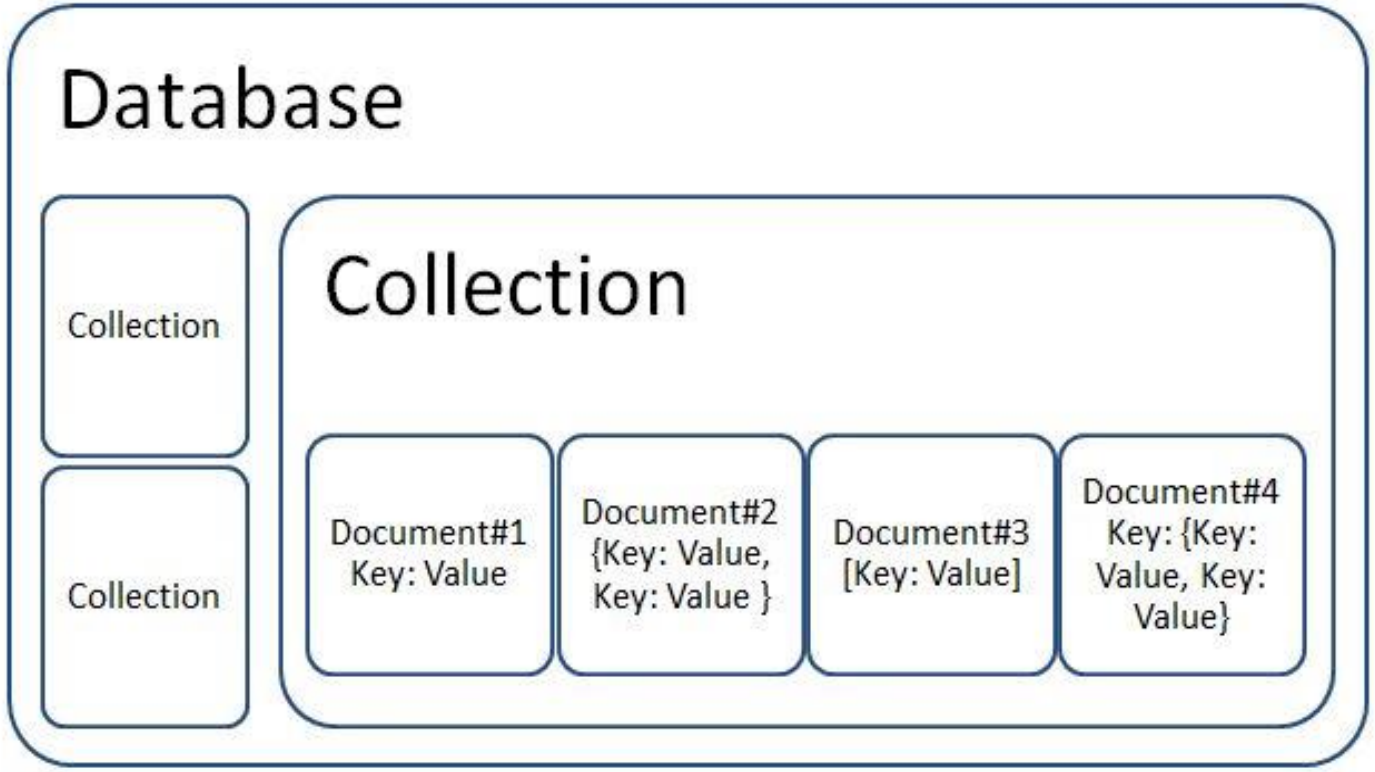
```
> set mykey hello
OK
> exists mykey
(integer) 1
> del mykey
(integer) 1
> exists mykey
(integer) 0
```

```
> set key 100 ex 10
OK
> ttl key
(integer) 9
```


```
> rpush mylist 1 2 3 4 5 "foo bar"
(integer) 9
> lrange mylist 0 -1
1) "first"
2) "A"
3) "B"
4) "1"
5) "2"
6) "3"
7) "4"
8) "5"
9) "foo bar"
```

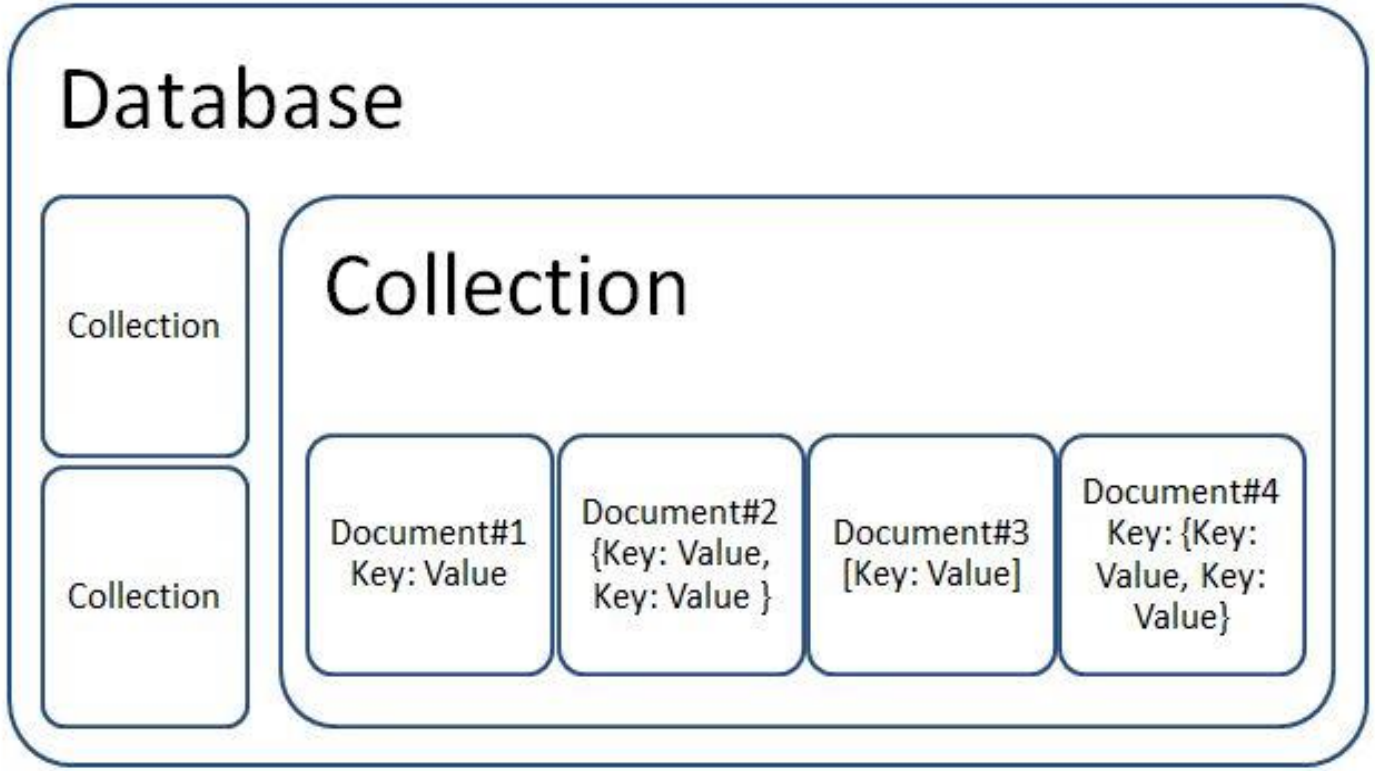
Document-oriented Collections

- Database contains many ***Collections***.
- Collections contain many ***documents***.
- Suitable for Web based applications and RESTful services.
- *E.g.* ***MongoDB, Couchbase***



Document-oriented Collections

- Offer support to store **semi-structured data**
- E.g. JSON, XML, … even a Word document
- Unit of data: **document** (similar to row in RDBMS)
- Table which contains a group of documents is called a "**Collection**"



```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
             phone: "123-456-7890",
             email: "xyz@example.com"
           },
  access: {
            level: 5,
            group: "dev"
          }
}
```

Embedded sub-document

Embedded sub-document

user document

```
{
  _id: <ObjectId1>,
  username: "123xyz"
}
```

contact document

```
{
  _id: <ObjectId2>,
  user_id: <ObjectId1>,
  phone: "123-456-7890",
  email: "xyz@example.com"
}
```

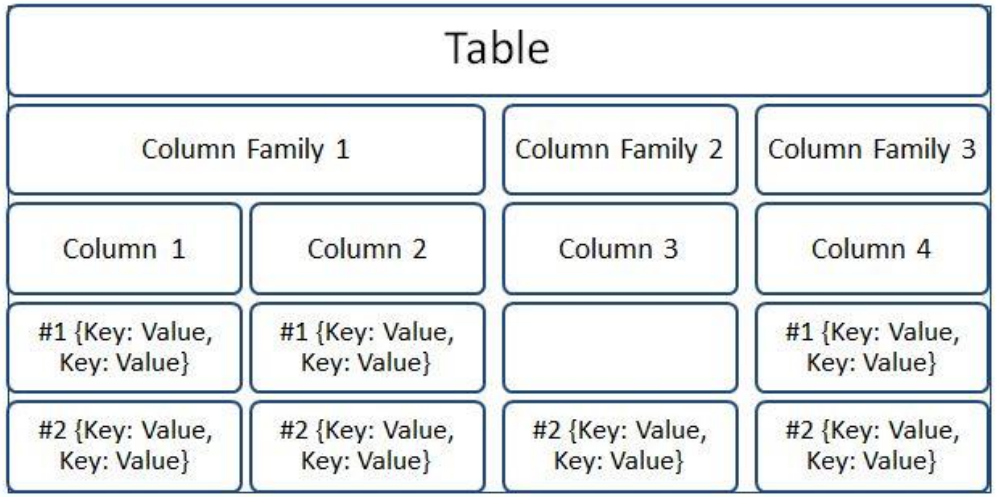
access document

```
{
  _id: <ObjectId3>,
  user_id: <ObjectId1>,
  level: 5,
  group: "dev"
}
```

Column-based Store

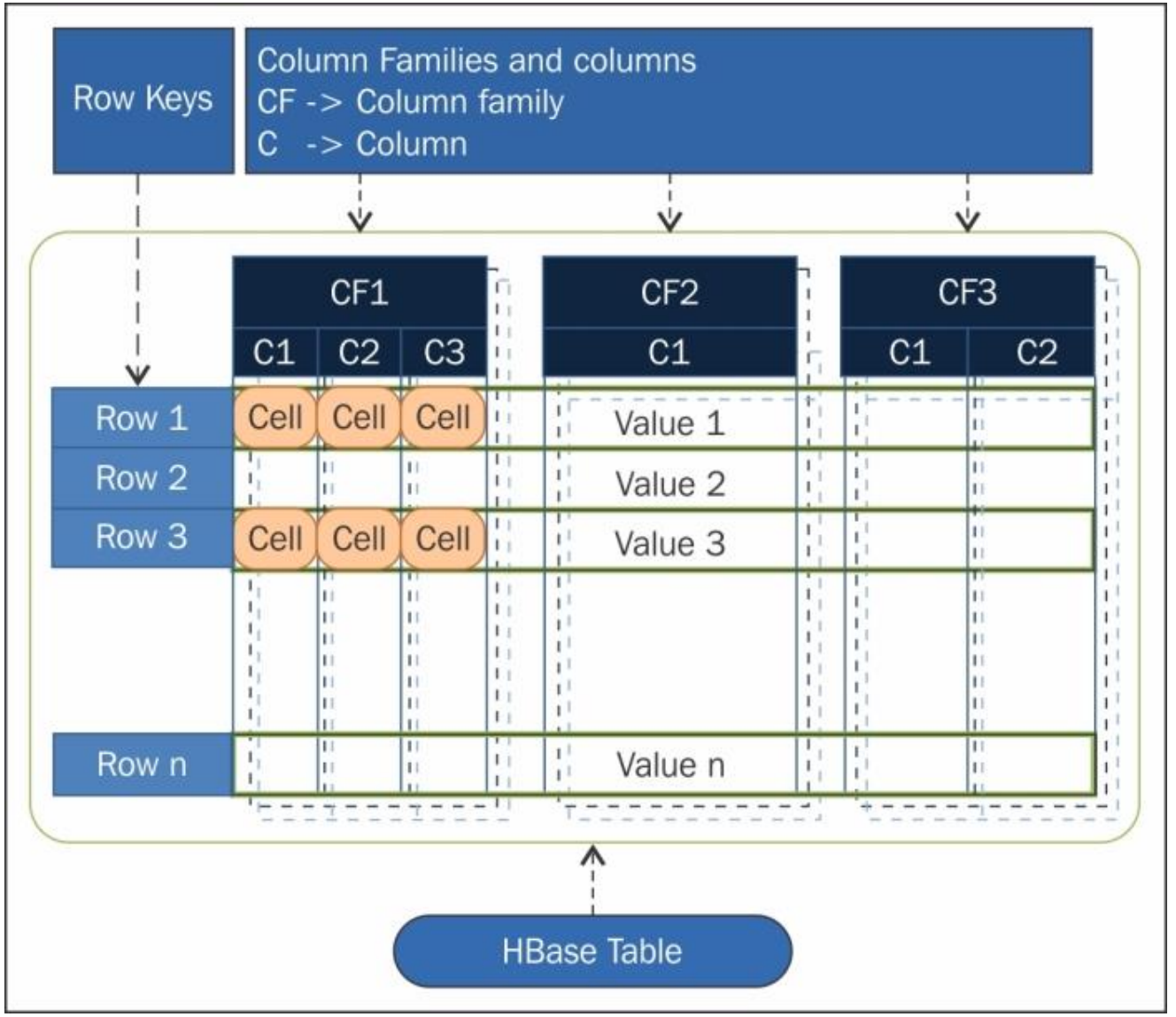
- Due to the wide table, it supports ***column families***.
- Each column family contains many (a group of) columns.
- Values for columns - ***sparsely distributed*** with key-value pairs.
- Suitable for Data warehousing and OLAP applications
- *E.g.* ***HBase, Cassandra***

Column oriented databases are developed based on the **Big Table** whitepaper published by **Google**.



This takes a different approach than traditional RDBMS, where it supports to add more and more columns resulting in wider tables.

APACHE HBASE



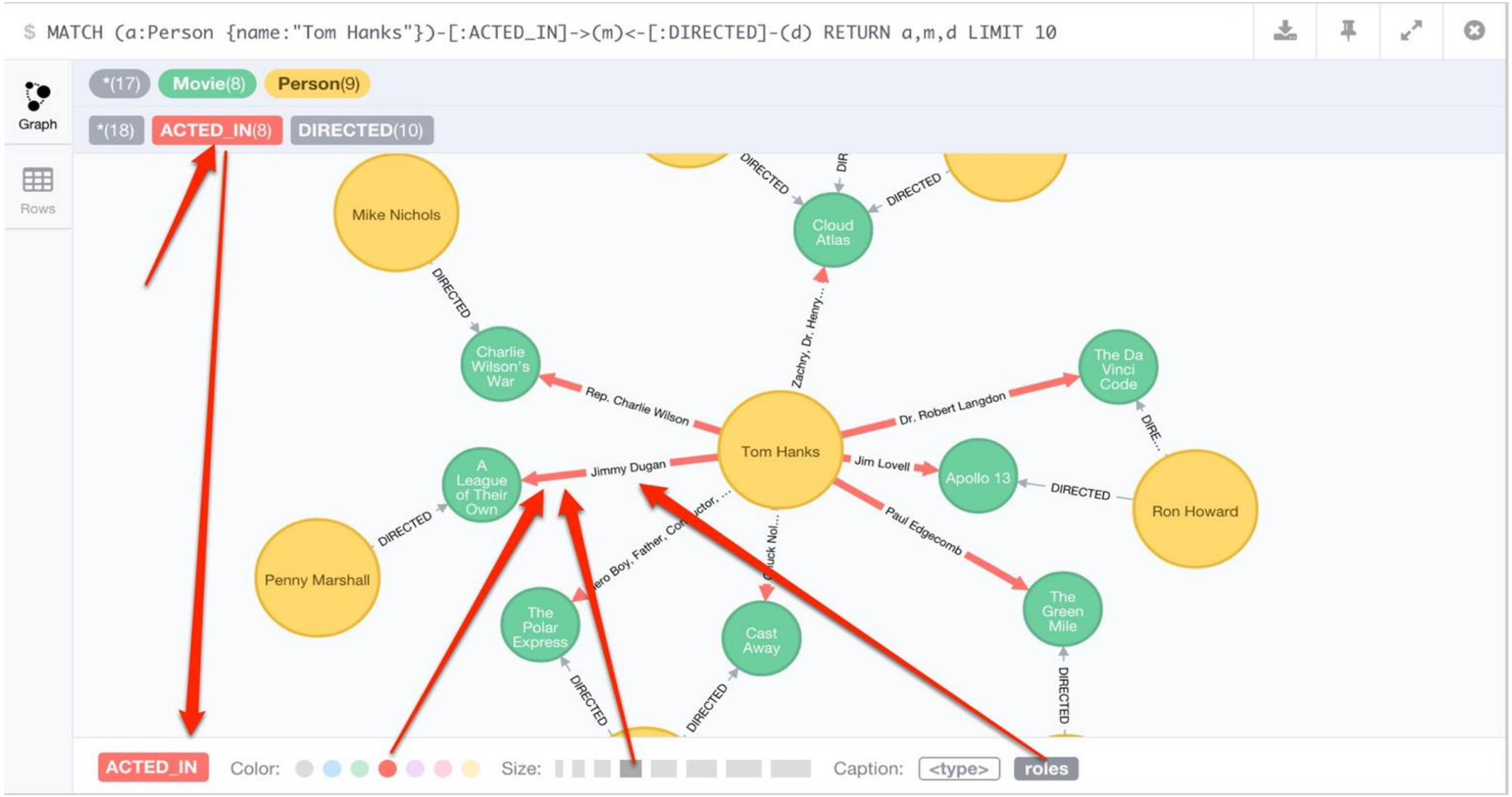
Graph-based Databases

- *2D representation of* ***graph***.
- A network of ***nodes*** and ***edges*** that represent and store data.
- Suitable for social media, network problems which involve complex queries with more joins.
- *E.g.* ***Neo4J, OrientDB, HyperGraphDB, GraphBase, IinfiniteGraph***

Real world graphs contain nodes and edges. In a graph database they are called nodes and relations. The graph database is the two dimensional representation of a graph (works better on *directed* graphs).

Graph			
Node	Labels / Properties	Relations	Relation Properties
Node Id A	{Key: Value, Key: Value, Key Value}	Node Id B	{Key: Value, Key: Value}
Node Id A	{Key, Value, Key: Value}	Node Id C	{Key: Value}
Node Id B	{Key: Value}	Node Id A	{Key: Value, Key: Value}

Each graph contains Node, Node Properties, Relation and Relation Properties as Columns. There will be values in each row for these columns. The values for properties columns can have key-value pairs.


$ MATCH (a:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:DIRECTED]-(d) RETURN a,m,d LIMIT 10
Graph
Rows
*(17)
Movie(8)
Person(9)
*(18)
ACTED_IN(8)
DIRECTED(10)
Mike Nichols
Cloud Atlas
Charlie Wilson's War
Rep. Charlie Wilson
Tom Hanks
Dr. Robert Langdon
The Da Vinci Code
Jimmy Dugan
A League of Their Own
Jim Lovell
Apollo 13
Ron Howard
Penny Marshall
Paul Edgecomb
The Polar Express
Cast Away
The Green Mile
DIRECTED
ACTED_IN
Color:
Size:
Caption:
<type>
roles

Feature Highlight

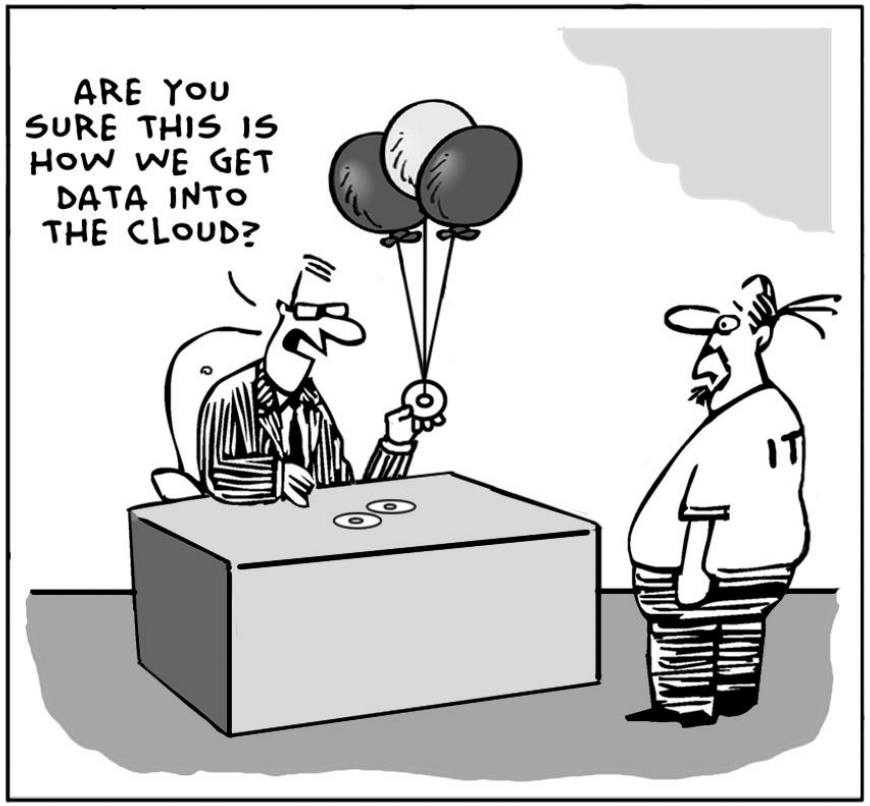
TYPES	PERFORMANCE	SCALABILITY	FLEXIBILITY	COMPLEXITY
KEY-VALUE STORE	high	high	high	none
COLUMN STORE	high	high	moderate	low
DOCUMENT	high	variable (high)	high	low
GRAPH DATABASE	variable	variable	high	high

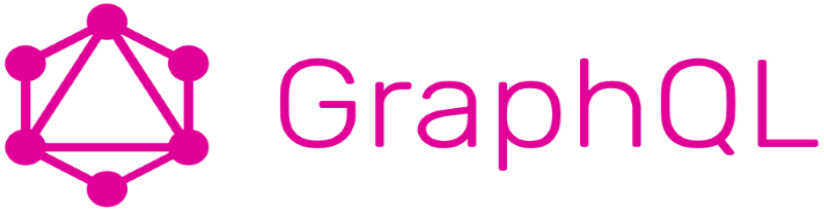
What Do You THINK??

When to use what?

#TheBigQuestion

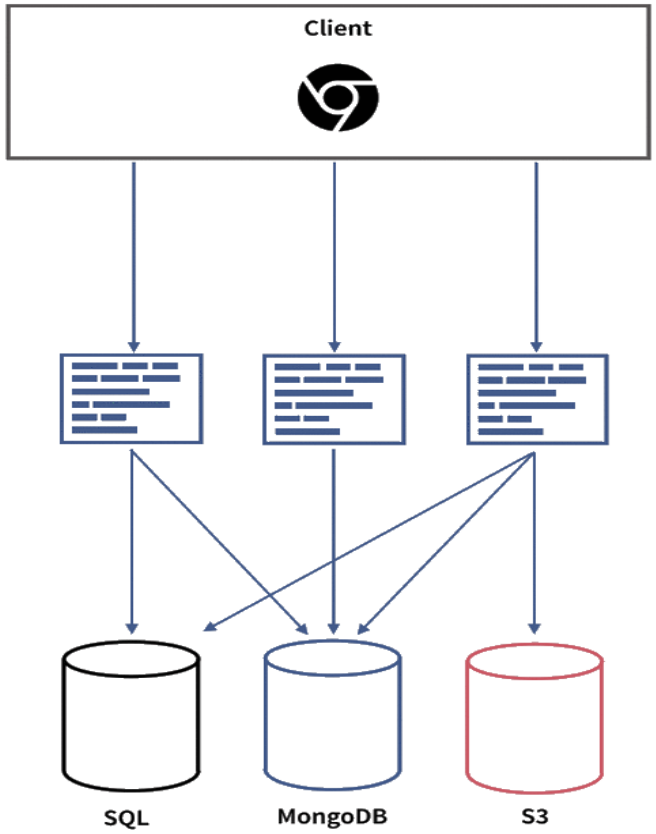
Given a set of data requirements, what factors would influence your decision about choosing an appropriate database technology?



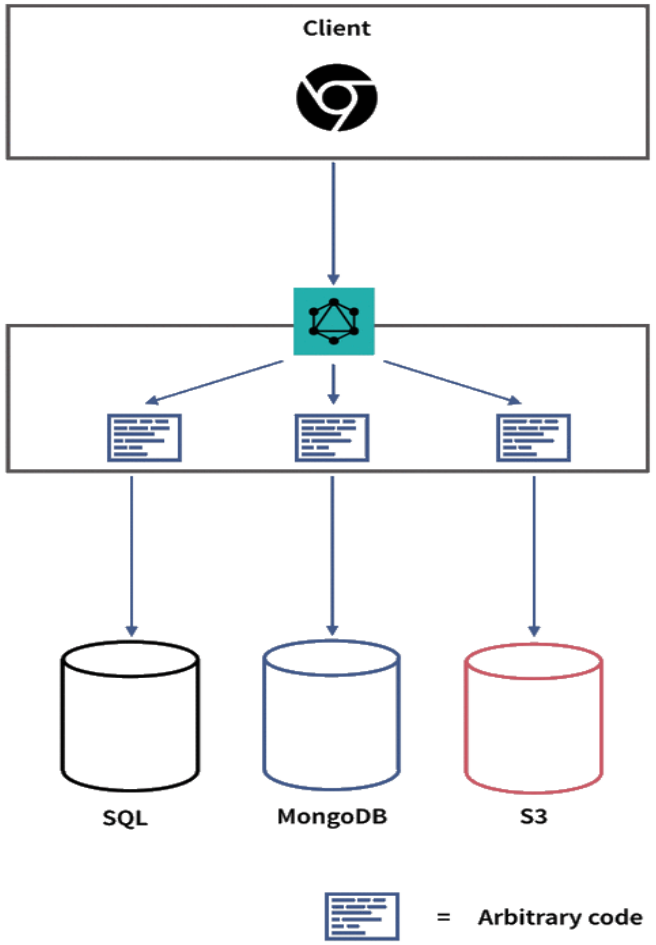


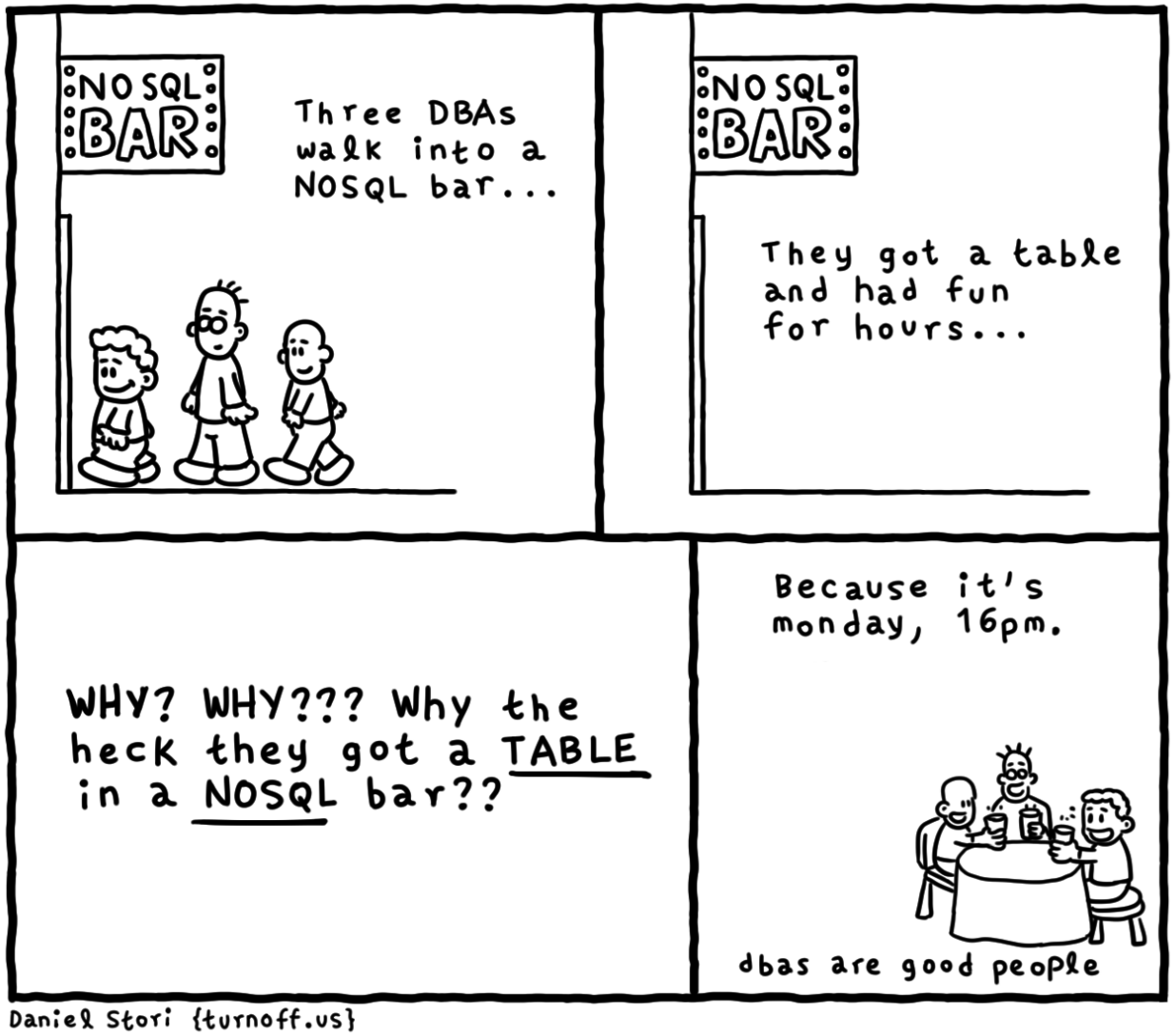
- Ask for what you need, get exactly that
- Get many resources in a single request
- Describe what's possible with a type system

REST



GRAPHQL




NO SQL BAR
Three DBAs walk into a NOSQL bar...
NO SQL BAR
They got a table and had fun for hours...
WHY? WHY??? Why the heck they got a TABLE in a NOSQL bar??
Because it's monday, 16pm.
dbas are good people
Daniel Stori {turnoff.us}

Thank You