RDBMS vs NoSQL

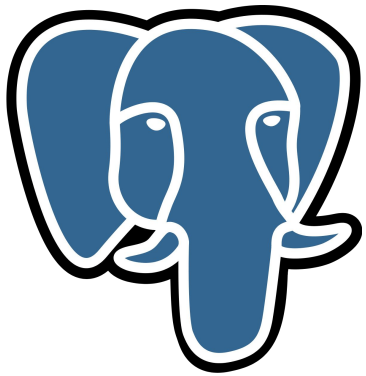
No index found

Relational Databases Management System

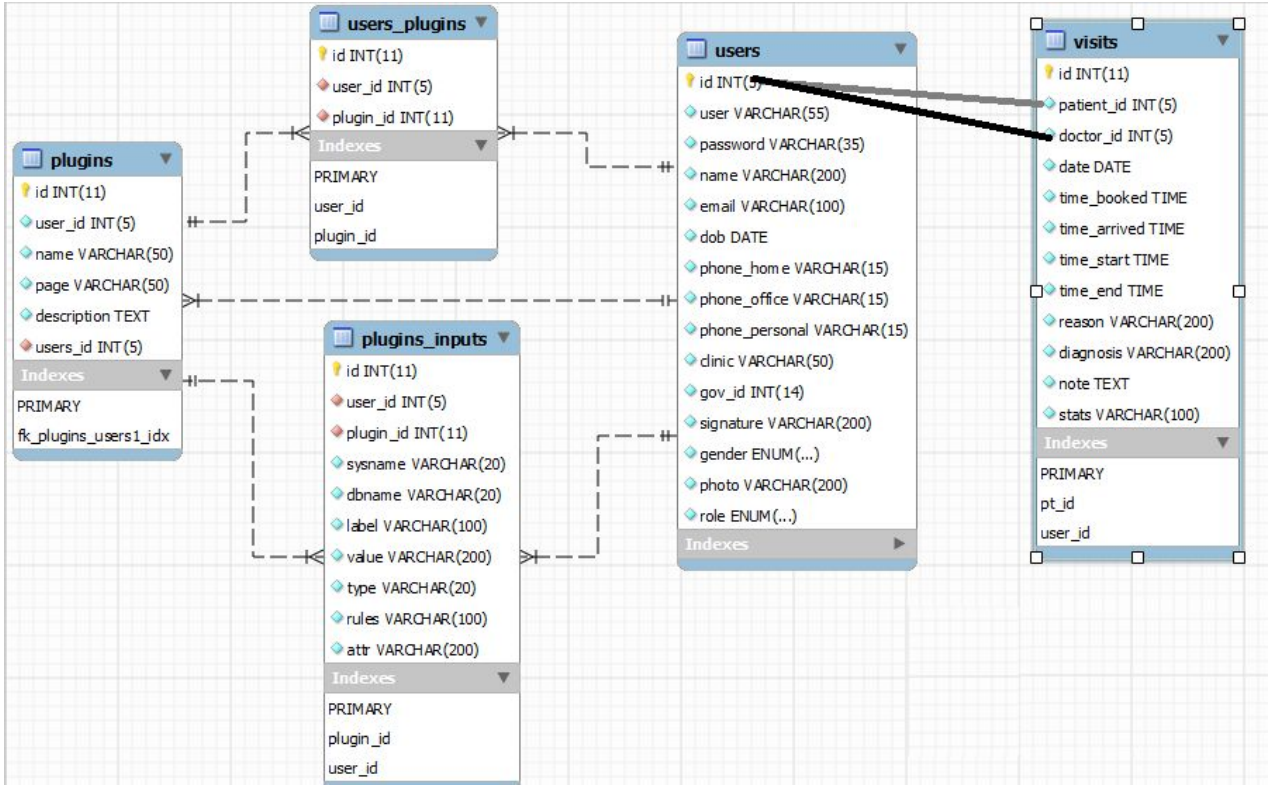
SQL databases represent data in form of tables which consists of n number of rows of data.

Predefined schema for table definition.

Use SQL command to query and aggregate.



Examples


users_plugins
id INT(11)
user_id INT(5)
plugin_id INT(11)
Indexes
PRIMARY
user_id
plugin_id
users
id INT(5)
user VARCHAR(55)
password VARCHAR(35)
name VARCHAR(200)
email VARCHAR(100)
dob DATE
phone_home VARCHAR(15)
phone_office VARCHAR(15)
phone_personal VARCHAR(15)
clinic VARCHAR(50)
gov_id INT(14)
signature VARCHAR(200)
gender ENUM(...)
photo VARCHAR(200)
role ENUM(...)
Indexes
visits
id INT(11)
patient_id INT(5)
doctor_id INT(5)
date DATE
time_booked TIME
time_arrived TIME
time_start TIME
time_end TIME
reason VARCHAR(200)
diagnosis VARCHAR(200)
note TEXT
stats VARCHAR(100)
Indexes
PRIMARY
pt_id
user_id
plugins
id INT(11)
user_id INT(5)
name VARCHAR(50)
page VARCHAR(50)
description TEXT
users_id INT(5)
Indexes
PRIMARY
fk_plugins_users1_idx
plugins_inputs
id INT(11)
user_id INT(5)
plugin_id INT(11)
sysname VARCHAR(20)
dbname VARCHAR(20)
label VARCHAR(100)
value VARCHAR(200)
type VARCHAR(20)
rules VARCHAR(100)
attr VARCHAR(200)
Indexes
PRIMARY
plugin_id
user_id

The problem with RDBMS

We need to defined a lot of table definitions.

```
CREATE TABLE table_name
(
     column1 datatype,
     column2 datatype,
     column3 datatype,
   ....
);
```

```
CREATE TABLE table_name
(
     column1 datatype,
     column2 datatype,
     column3 datatype,
   ....
);
```

```
CREATE TABLE table_name
(
     column1 datatype,
     column2 datatype,
     column3 datatype,
   ....
);
```

The problem with RDBMS

Pain in the ass to do query later. Joining problem will come after all.

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate

FROM Orders

INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

The problem with RDBMS

Rigid schema. We are on the age of data!

The problem with RDBMS

Rigid schema. We are on the age of data!

Semi or totally unstructured data is everywhere!

The problem with RDBMS

Rigid schema. We are on the age of data!

Semi or totally unstructured data is everywhere!

We have IOT, enterprise millennial documents!

The problem with RDBMS

Rigid schema. We are on the age of data!

Semi or totally unstructured data is everywhere!

We have IOT, enterprise millennial documents!

JSON EVERYWHERE!

NoSQL

non-relational or distributed database.

collection of key-value pair, documents, graph databases or wide-column stores.

Dynamic definition of schema, sometimes it called mapping.

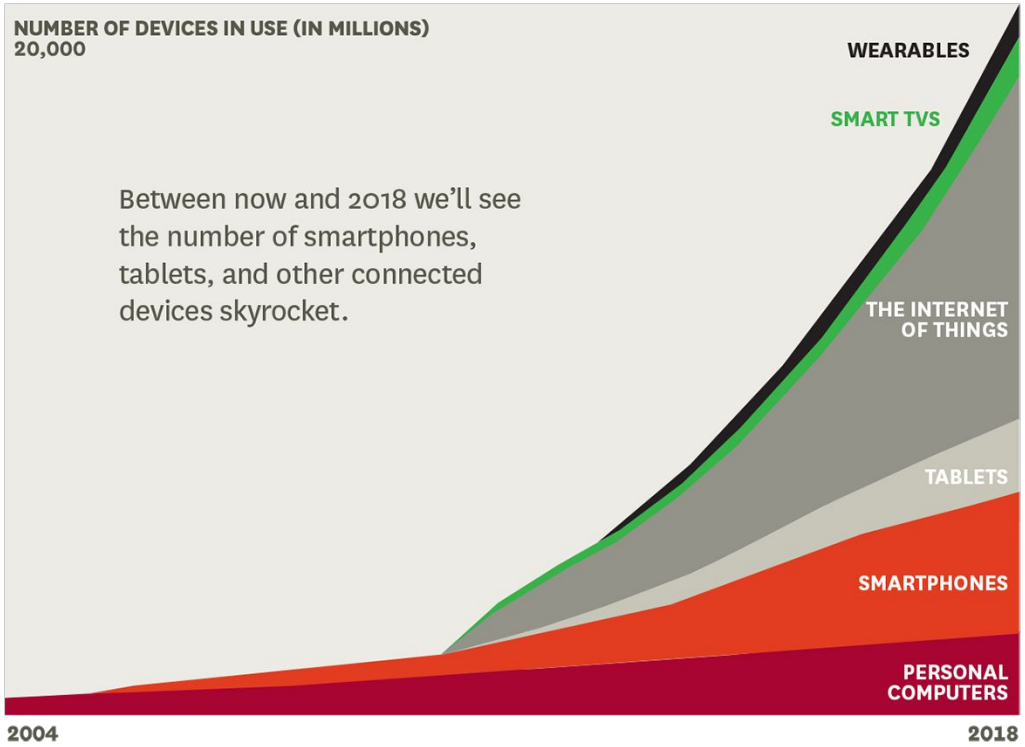
No standardized language for Query and Aggregate commands.

Why NoSQL?

Data boom!

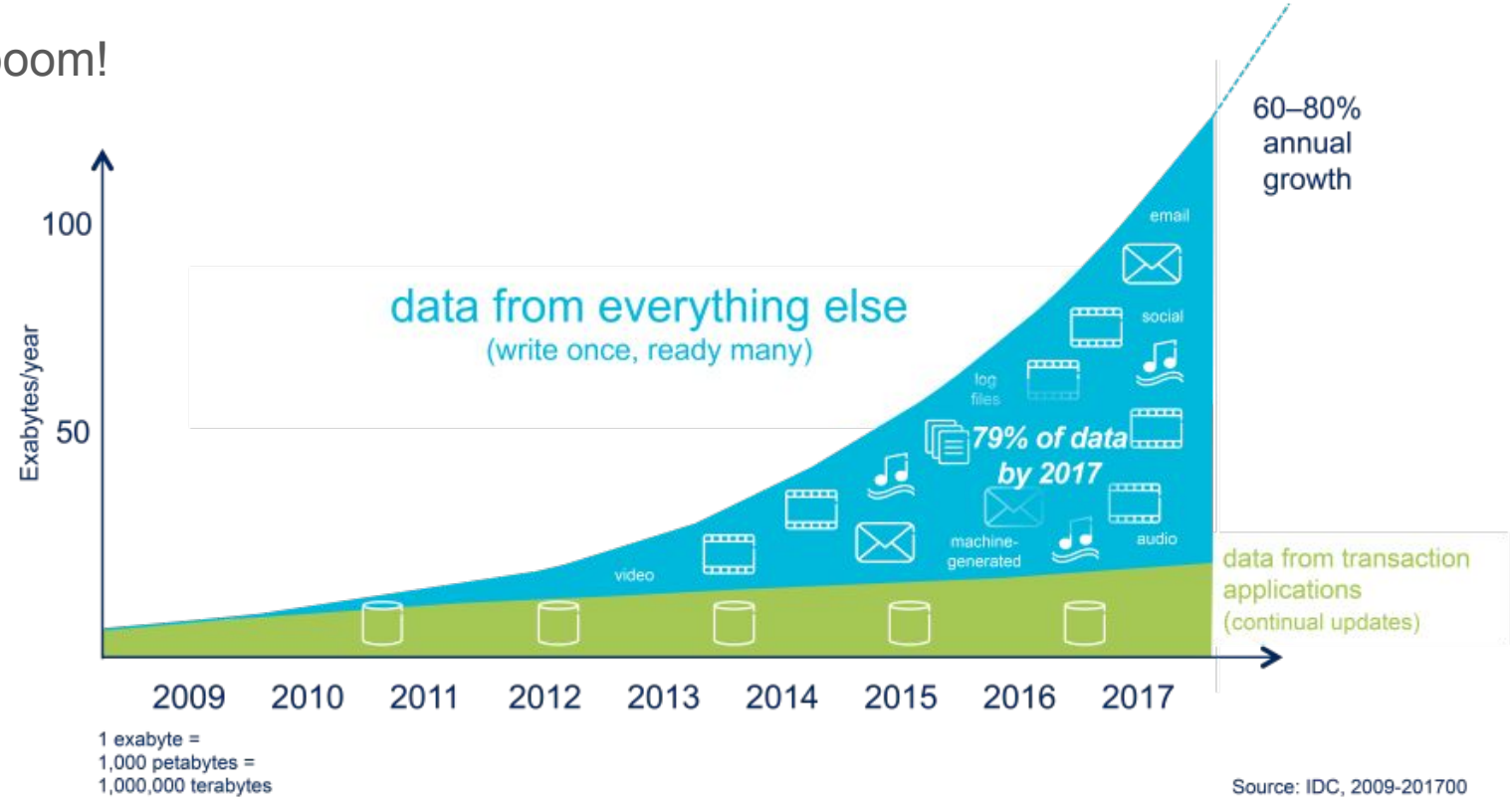
Why NoSQL?

Data boom!



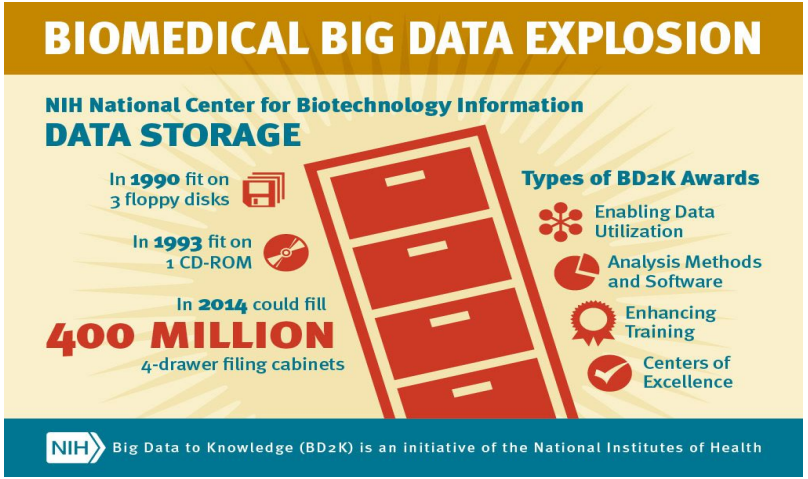
Why NoSQL?

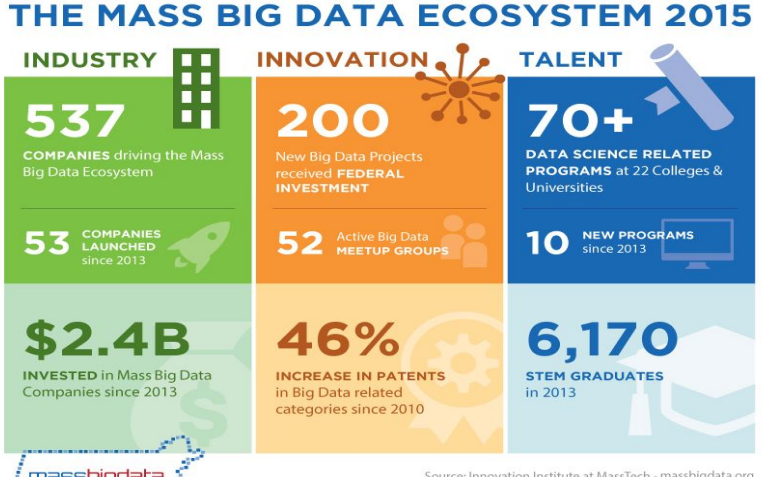
Data boom!



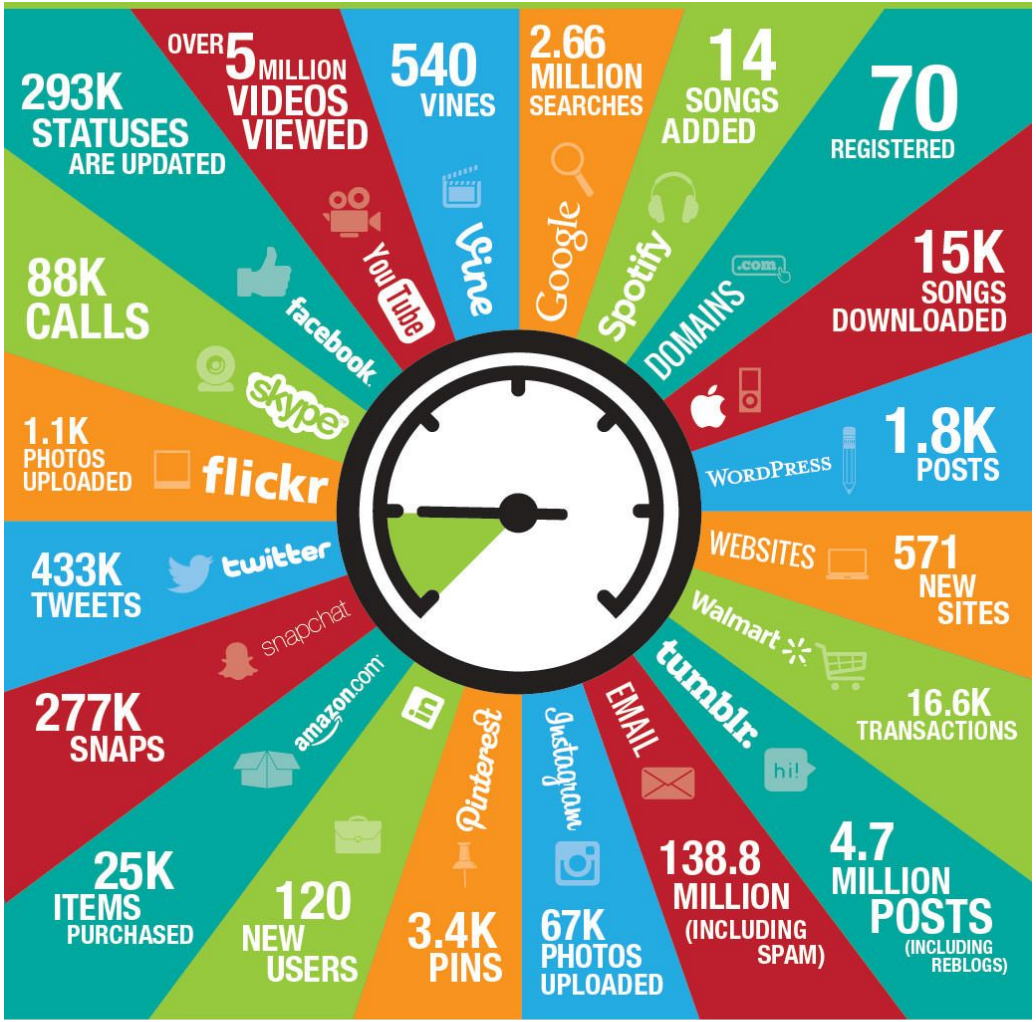
Why NoSQL?

Now literally everything can be represented as digital data!





In a minute,


293K STATUSES ARE UPDATED
OVER 5 MILLION VIDEOS VIEWED
540 VINES
2.66 MILLION SEARCHES
14 SONGS ADDED
70 REGISTERED
88K CALLS
facebook
YouTube
Vine
Google
Spotify
DOMAINS
.com
15K SONGS DOWNLOADED
skype
1.1K PHOTOS UPLOADED
flickr
WORDPRESS
1.8K POSTS
433K TWEETS
twitter
WEBSITES
571 NEW SITES
snapchat
Walmart
277K SNAPS
amazon.com
tumblr.
16.6K TRANSACTIONS
hi!
EMAIL
Instagram
Pinterest
25K ITEMS PURCHASED
120 NEW USERS
3.4K PINS
67K PHOTOS UPLOADED
138.8 MILLION (INCLUDING SPAM)
4.7 MILLION POSTS (INCLUDING REBLOGS)

Why NoSQL?

Again, JSON is everywhere.

Why NoSQL?

Again, JSON is everywhere.

Semi or totally unstructured data is everywhere.

Why NoSQL?

Again, JSON is everywhere.

Semi or totally unstructured data is everywhere.

AND WE WANT TO STORE EVERYTHING!

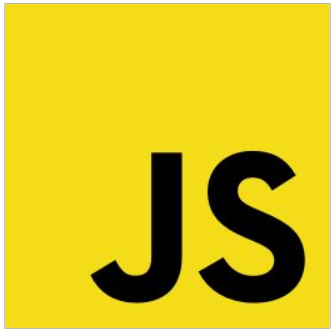
Why NoSQL?

Rise of dynamic programming language.

Why NoSQL?

Rise of dynamic programming language.

Python, Javascript, Ruby, Groovy and etc.

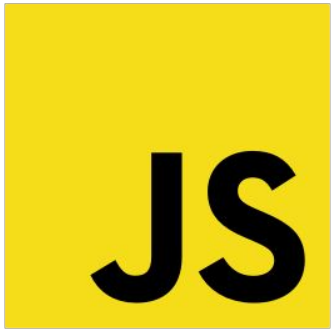


Why NoSQL?

Rise of dynamic programming language.

Python, Javascript, Ruby, Groovy and etc.

We don’t even know what is the type of a variable!

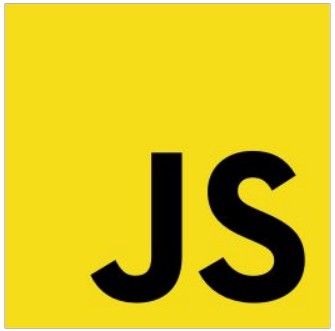


Why NoSQL?

Rise of dynamic programming language.

Python, Javascript, Ruby, Groovy and etc.

We don’t even know what is the type of a variable!
And we don’t even care!



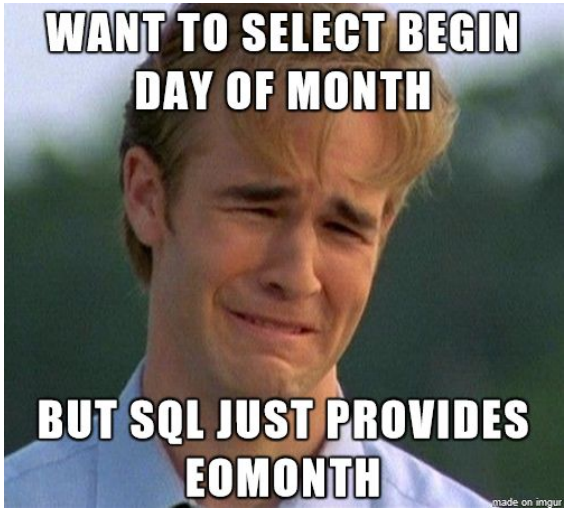
Why NoSQL?

NoSQL makes sense on applications utilized very large unstructured data with less pain in the ass to filter.

Why NoSQL?

NoSQL makes sense on applications utilized very large unstructured data with less pain in the ass to filter.

Suitable for logging, timestamp data, IOT data.



Type of NoSQL

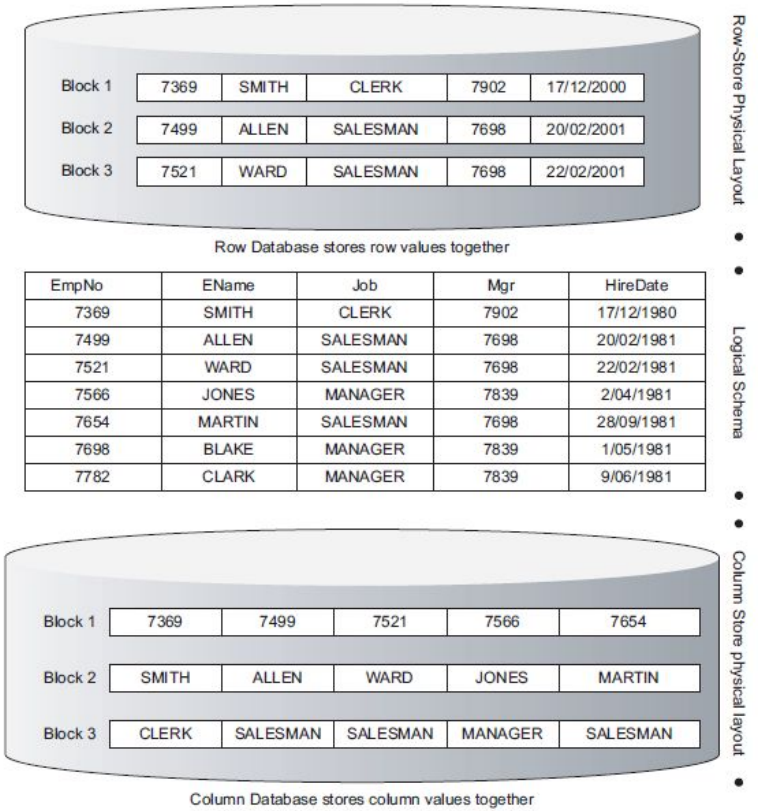
Column based.

Type of NoSQL

Column based.

It stores data as column wise and each row has many columns.

Example, Cassandra, DynamoDB.



Type of NoSQL

Document based.

Type of NoSQL

Document based.

Hierarchical tree structured mapping.

The famous one we often use it.

Example, MongoDB, Elastic Search.

JSON DOCUMENT

Type of NoSQL

Graph based.

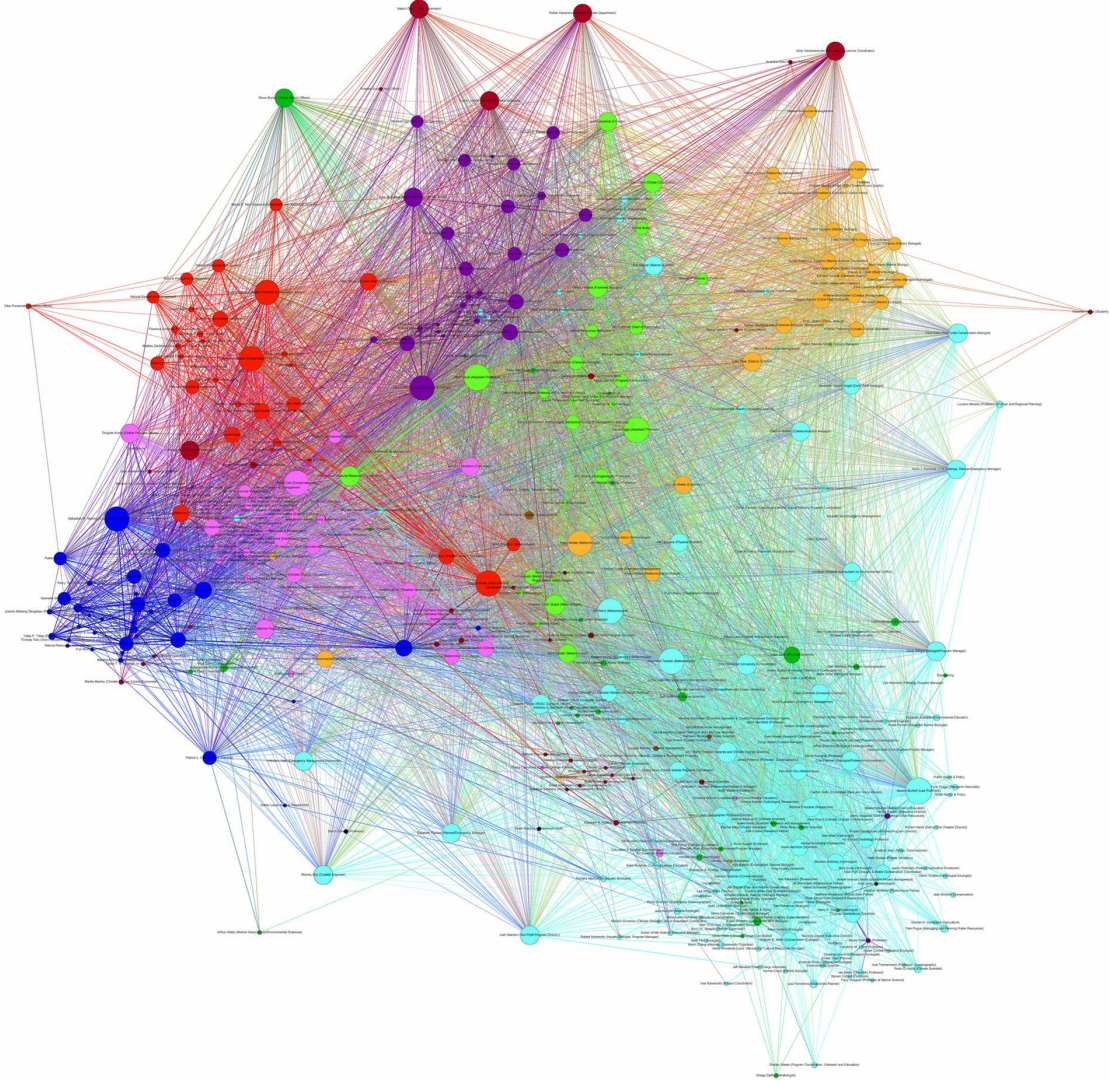
Type of NoSQL

Graph based.

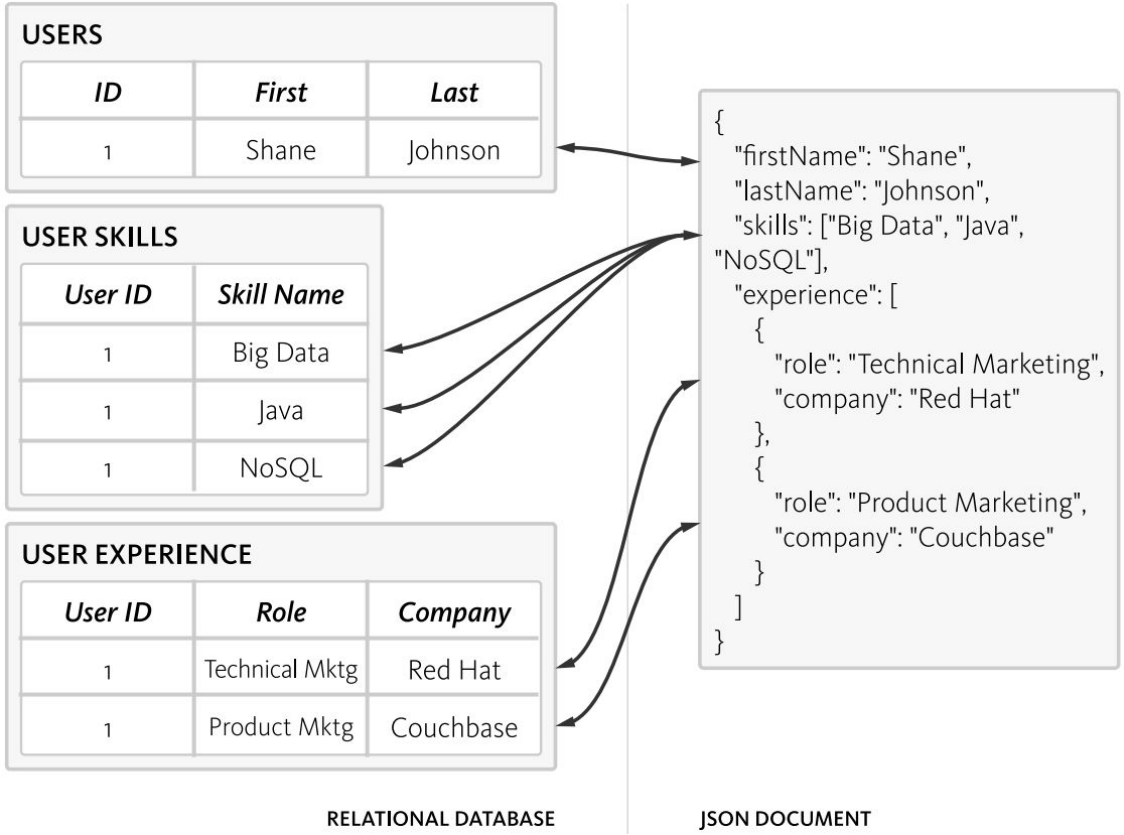
Stores entities and relationship as nodes and edges.

Suitable for social network analysis.

Example, neo4j



SQL vs NoSQL


USERS
ID
First
Last
1
Shane
Johnson
USER SKILLS
User ID
Skill Name
1
Big Data
1
Java
1
NoSQL
USER EXPERIENCE
User ID
Role
Company
1
Technical Mktg
Red Hat
1
Product Mktg
Couchbase
{
"firstName": "Shane",
"lastName": "Johnson",
"skills": ["Big Data", "Java", "NoSQL"],
"experience": [
{
"role": "Technical Marketing",
"company": "Red Hat"
},
{
"role": "Product Marketing",
"company": "Couchbase"
}
]
}
RELATIONAL DATABASE
JSON DOCUMENT

SQL vs NoSQL

Relational Model

Document Model

```
{
  "_id" : "rp-prod132546",
  "name" : "Marvel T2 Athena",
  "brand" : "Pinarello",
  "category" : "bike",
  "type" : "Road Bike",
  "price" : 2949.99,


  "size" : "55cm",
  "wheel_size" : "700c",
  "frameset" : {
     "frame" : "Carbon Toryaca",
     "fork" : "Onda 2V C"
  },
  "groupset" : {
     "chainset" : "Camp. Athena 50/34",
     "brake" : "Camp."
  },
  "wheelset" : {
     "wheels" : "Camp. Zonda",
     "tyres" : "Vittoria Pro"
  }
}
```

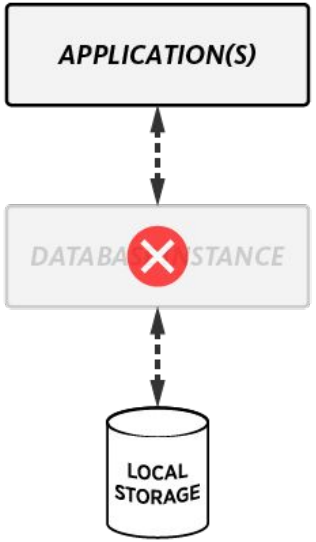
Relational Model

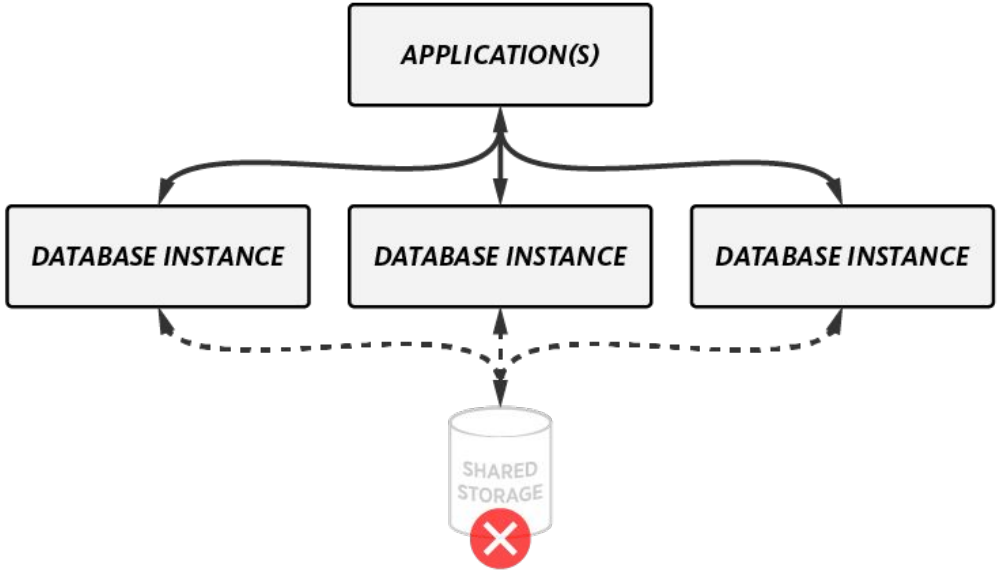
```
SELECT * FROM (
    SELECT
        ce.sku,
        ea.attribute_id,
        ea.attribute_code,
        CASE ea.backend_type
            WHEN 'varchar' THEN ce_varchar.value
            WHEN 'int' THEN ce_int.value
            WHEN 'text' THEN ce_text.value
            WHEN 'decimal' THEN ce_decimal.value
            WHEN 'datetime' THEN ce_datetime.value
            ELSE ea.backend_type
        END AS value,
        ea.is_required AS required
    FROM catalog_product_entity AS ce
    LEFT JOIN eav_attribute AS ea
        ON ce.entity_type_id = ea.entity_type_id
    LEFT JOIN catalog_product_entity_varchar AS ce_varchar
        ON ce.entity_id = ce_varchar.entity_id
        AND ea.attribute_id = ce_varchar.attribute_id
        AND ea.backend_type = 'varchar'
  LEFT JOIN catalog_product_entity_text AS ce_text
        ON ce.entity_id = ce_text.entity_id
        AND ea.attribute_id = ce_text.attribute_id
        AND ea.backend_type = 'text'
    LEFT JOIN catalog_product_entity_decimal AS ce_decimal
        ON ce.entity_id = ce_decimal.entity_id
        AND ea.attribute_id = ce_decimal.attribute_id
        AND ea.backend_type = 'decimal'
    LEFT JOIN catalog_product_entity_datetime AS ce_datetime
        ON ce.entity_id = ce_datetime.entity_id
        AND ea.attribute_id = ce_datetime.attribute_id
        AND ea.backend_type = 'datetime'
    WHERE ce.sku = 'rp-prod-1234'
  ) AS tab
  WHERE tab.value != '';
```

Document Model

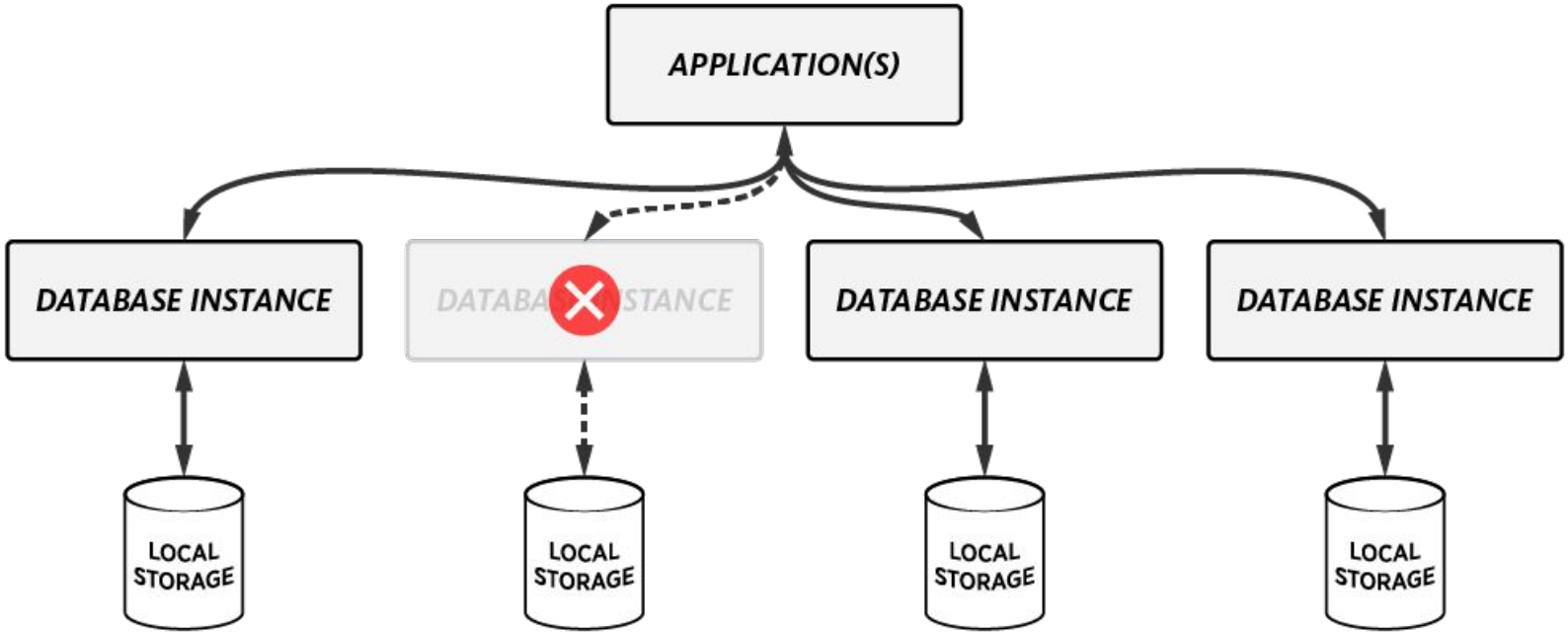
```
products.findById("rp-prod-1234")
```

SQL vs NoSQL, this one is SQL





SQL vs NoSQL, this one is NoSQL



SQL vs NoSQL