AT82.02

DATA MODELING AND MANAGEMENT

UNIT 2-1: NOSQL DATA MODEL AND MANAGEMENT

CHUTIPORN ANUTARIYA (CHUTI AT AIT DOT AC DOT TH)

Co-funded by the Erasmus+ Programme of the European Union

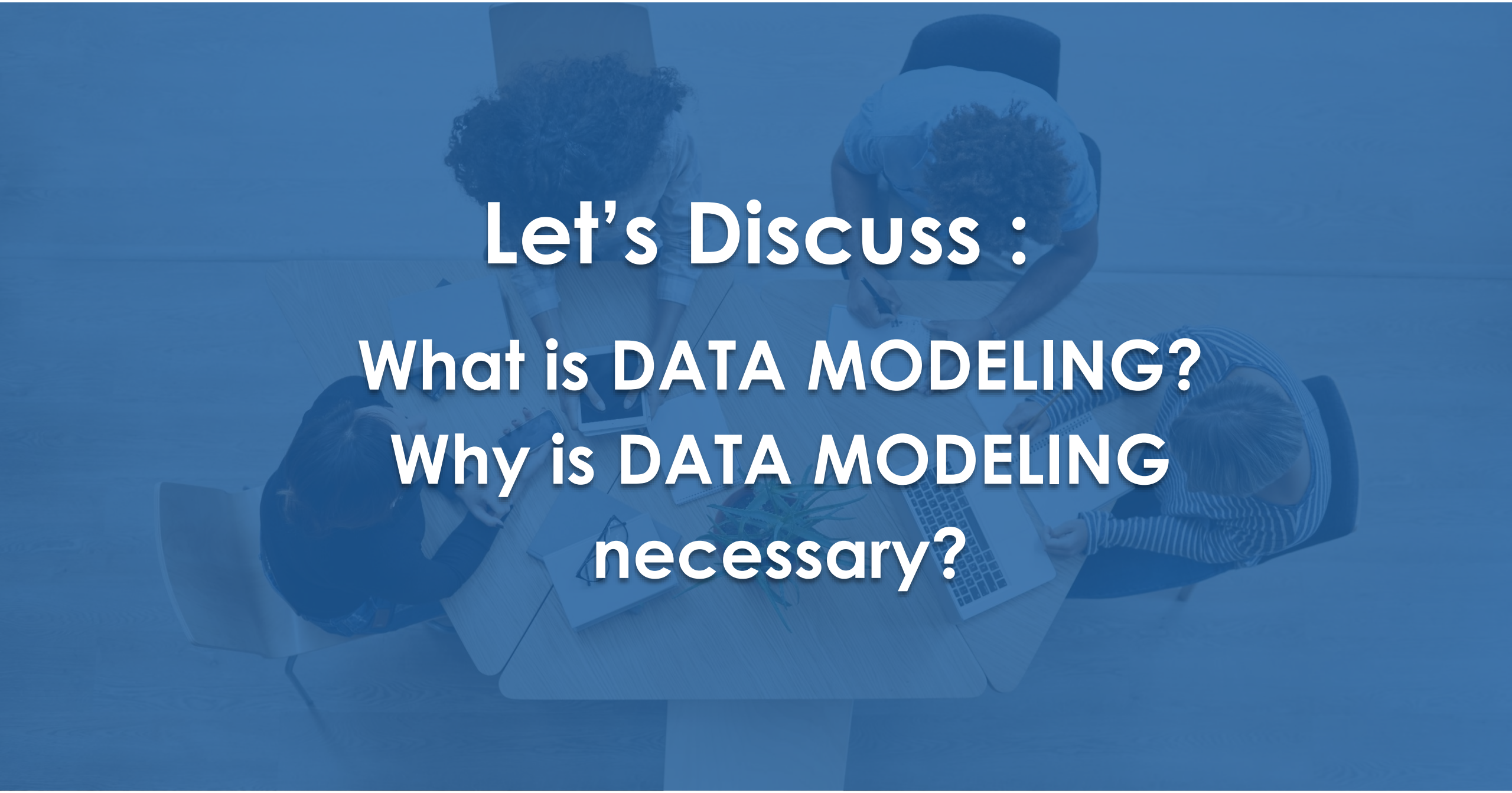
DS&AI

Why is DATA IMPORTANT?

Decision Making and Planning

Business Operation

Strategic Development



Let's Discuss :

What is DATA MODELING?
Why is DATA MODELING necessary?

What is DATA MODELING?
Why is DATA MODELING necessary?

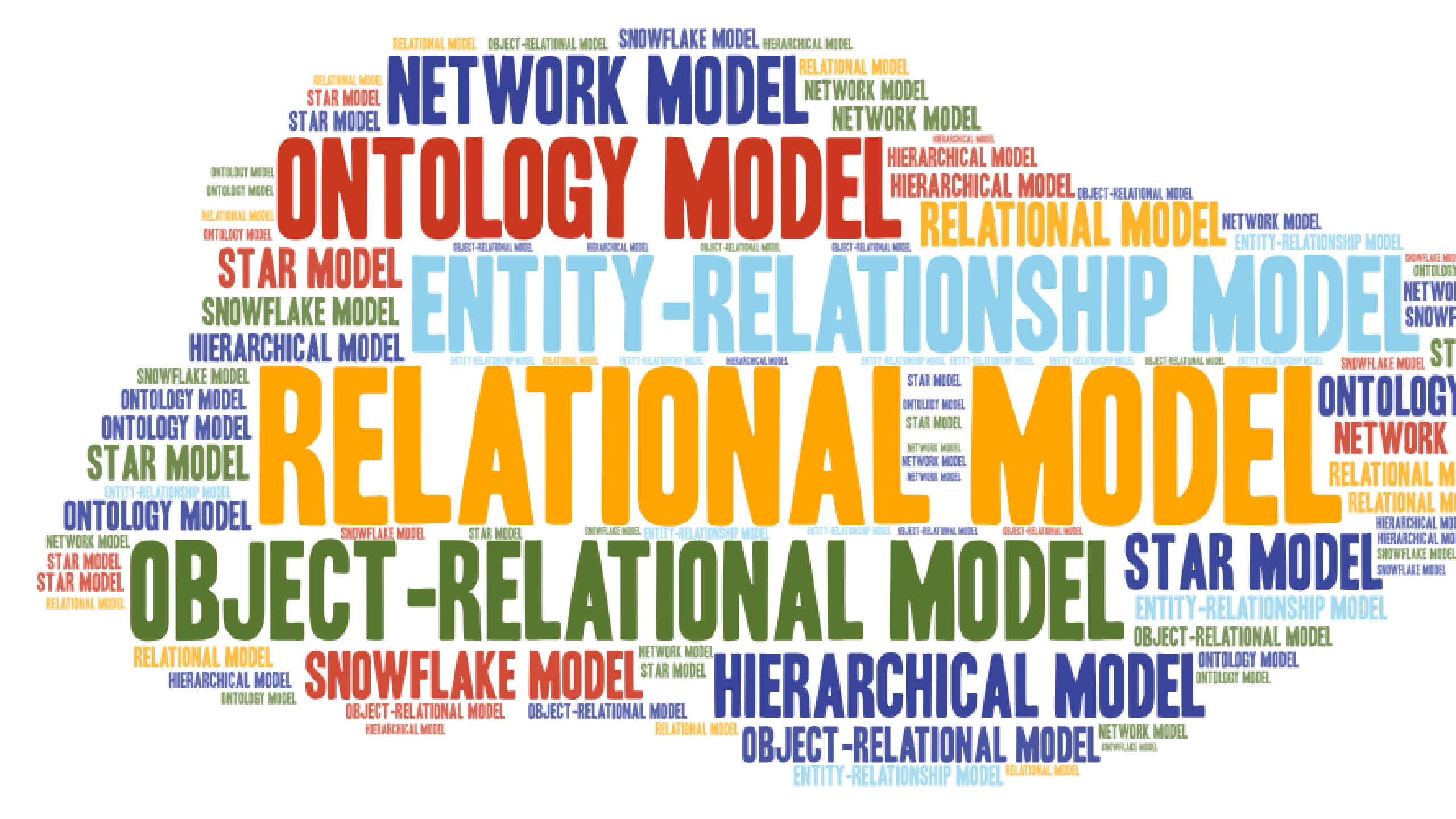
Large amounts of data imply a system or method to keep everything in order. The process of sorting and storing data is called "data modeling". A data model is a method by which we can organize and store data.

Proper models and storage environments offer the following benefits to large data:

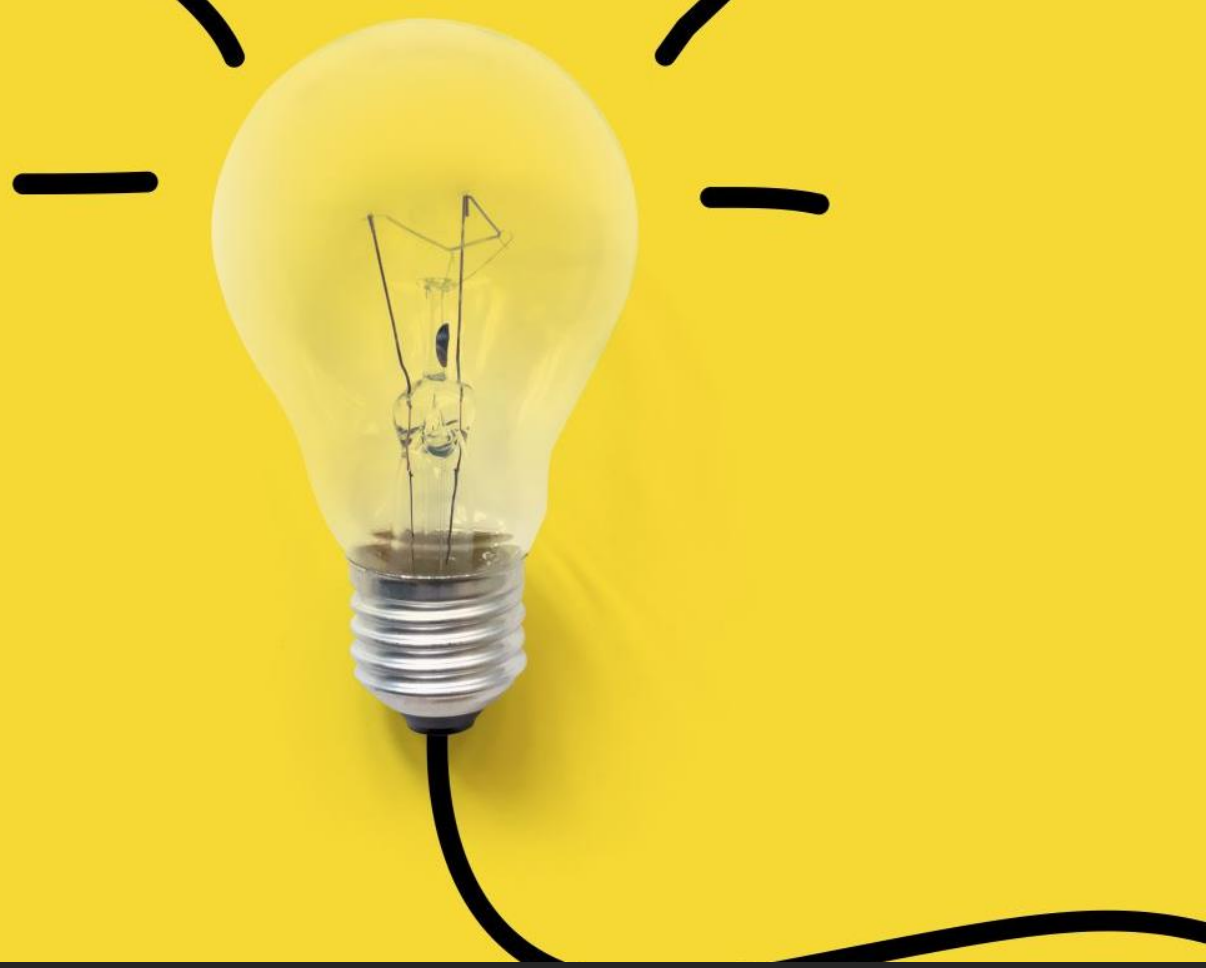
- **Performance:** Ensures fast query and reduces I/O output.
- **Cost:** Significantly reduces data redundancy, reducing storage and computing costs for the large data system.
- **Efficiency:** They greatly improve the user experience as well as the efficiency of data use.
- **Quality:** They make data statistics more consistent and reduce the possibility of computing errors.

Different Kinds of Data Models

LIST SOME DATA MODELS YOU ARE AWARE OR HAVE HEARD OF?

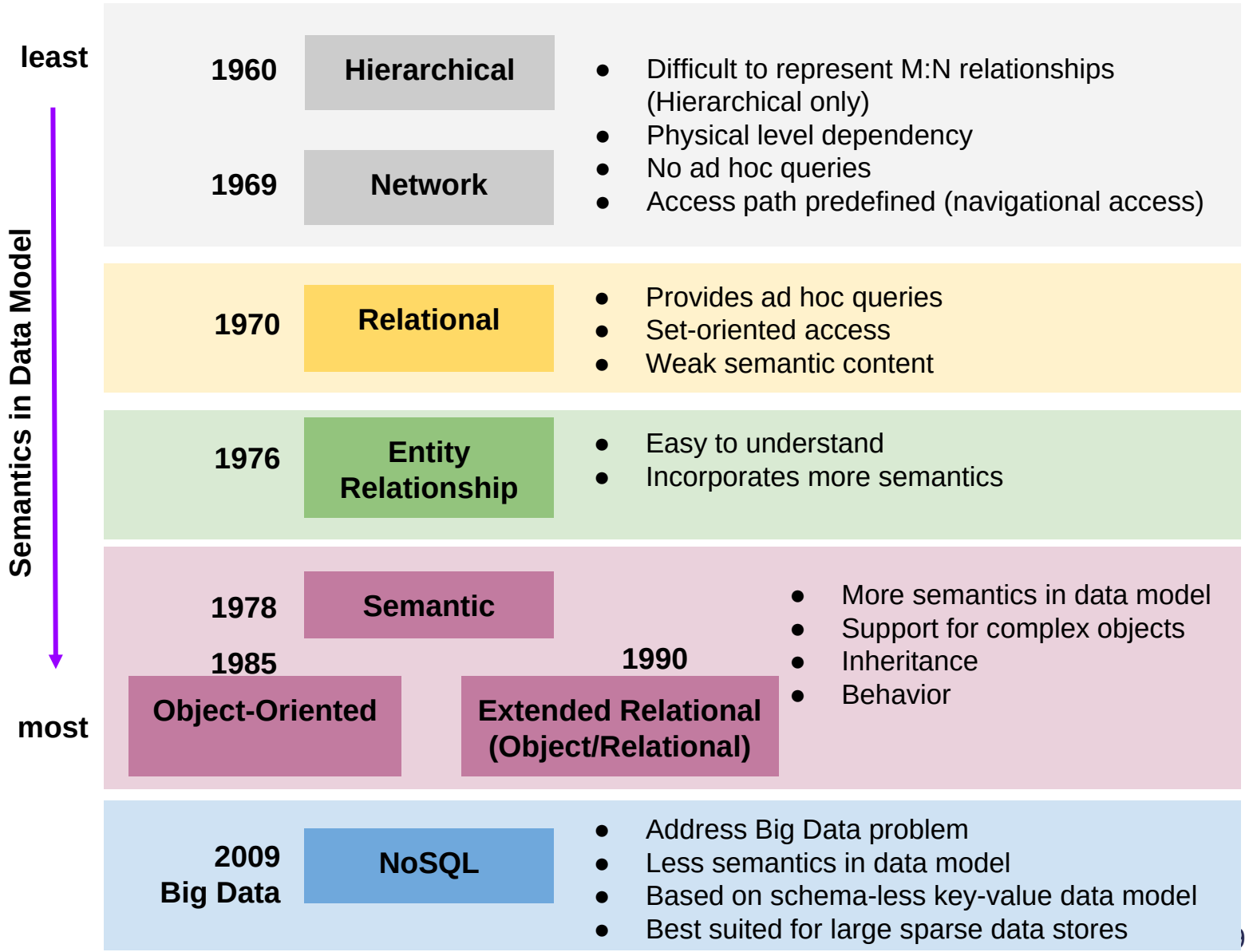

RELATIONAL MODEL
OBJECT-RELATIONAL MODEL
SNOWFLAKE MODEL
HIERARCHICAL MODEL
NETWORK MODEL
STAR MODEL
ONTOLOGY MODEL
ENTITY-RELATIONSHIP MODEL

Relational DBs are the most successful technology for the last 50 years



Fun Quiz: Relational & NoSQL

Evolution of data models



Relational

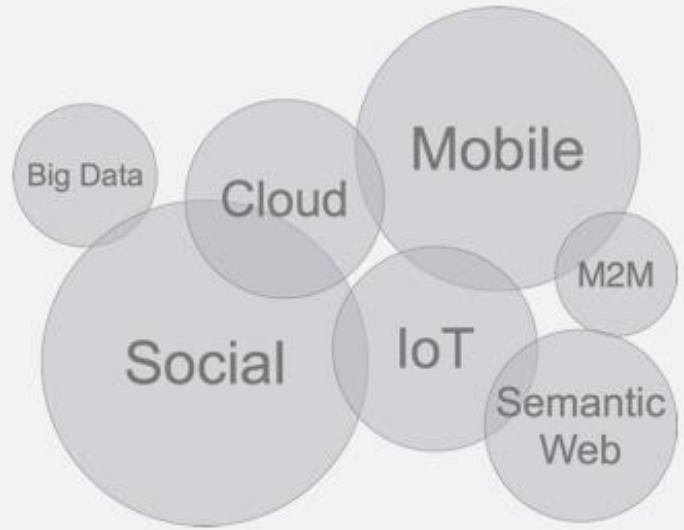
IBM Research Defines the Relational Database

Until the mid-1970s, computers sorted information using rigid, one-off database programs. Predecessor systems like IBM's IMS and VSAM on the mainframe could store megabytes of data, but it had to be entered and retrieved in the same structured way every time. IBM researcher E. F. "Ted" Codd wanted to improve the way data was sorted and handled. He sought to create a generalized description of how to store, update and extract data with accuracy, and query responses so any changes to data produced consistent results. In 1970, Codd completed his definition of the relational database, which became the foundation for IBM DB2 products.

NoSQL

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means **other than the tabular relations** used in relational databases. Motivations for this approach include: simplicity of design, "horizontal" scaling, which is a problem for relational databases, and finer control over availability

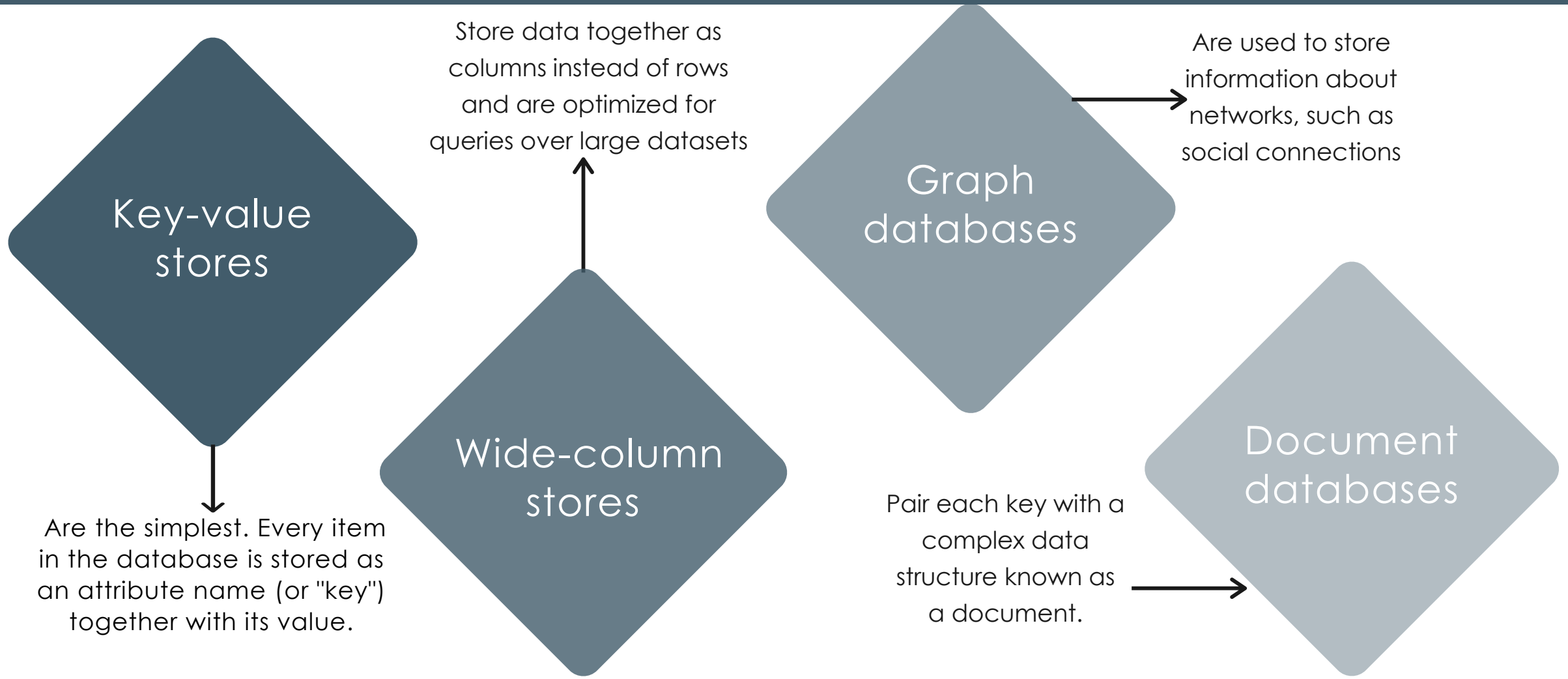
Structured Data	→	Unstructured Data
Small Datasets	→	Large Volume
Few Relationships	→	Connected Data
Waterfall Approach	→	Agile Approach
Scale Up	→	Scale Out
CIO	→	Developers



1970 2009 What's Next?

Major Categories of NOSQL Data Models

MAJOR CATEGORIES OF NOSQL DATA MODELS



DBMS Quadrant

One size fits all DB may not exist!

Polyglot Persistence

When storing data, it is best to use multiple data storage technologies, chosen based upon the way data is being used by individual applications or components of a single application.

Different kinds of data are best dealt with different data stores.

In short, it means picking the right tool for the right use case.

Polyglot Persistence example

An e-commerce platform will deal with many types of data (i.e. shopping cart, inventory, completed orders, etc) using a mixture of RDBMS solutions with NoSQL solutions

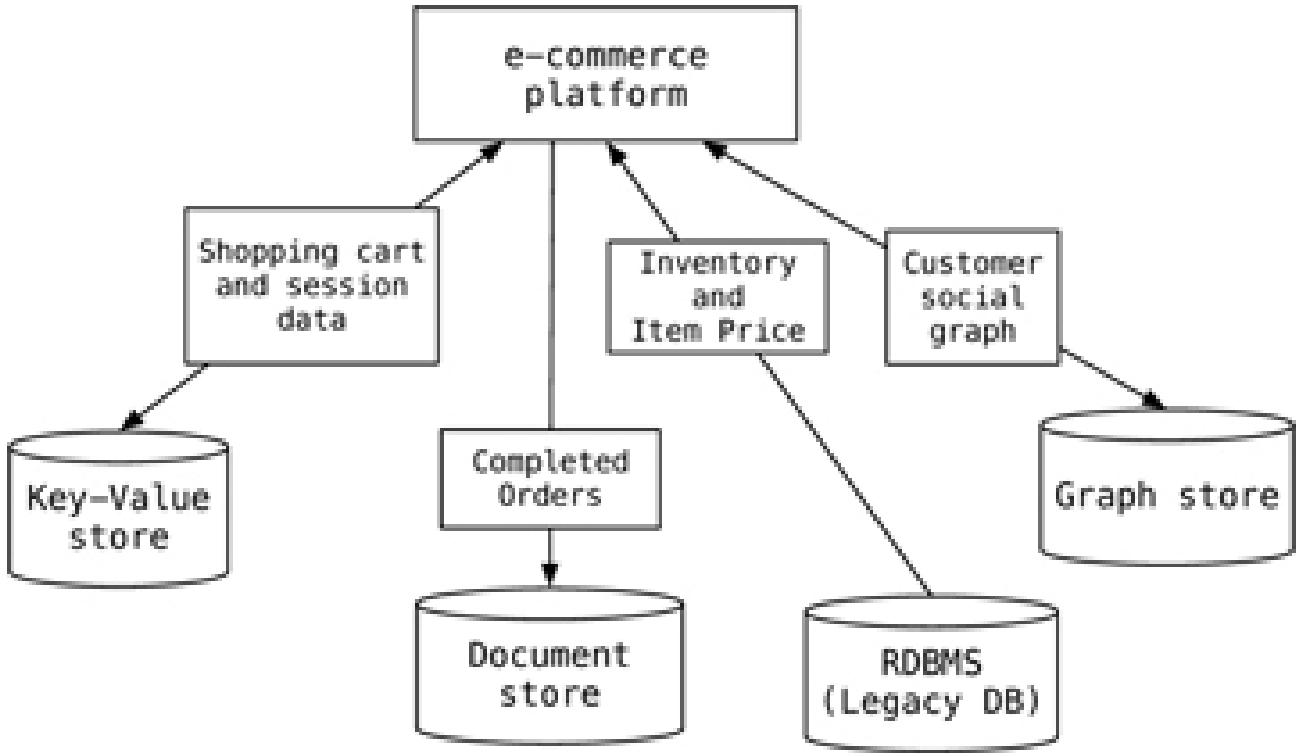


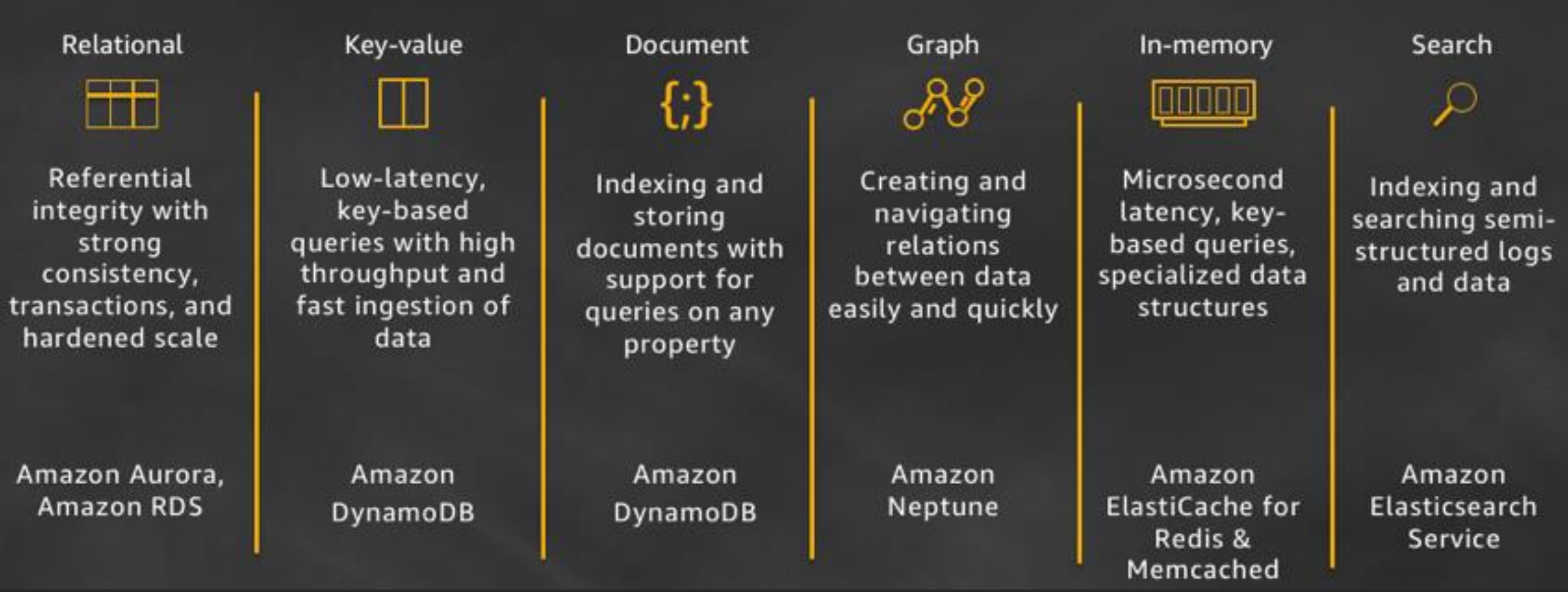
Figure 13.3. *Example implementation of polyglot persistence*

[src: https://www.jamesserra.com/archive/2015/07/what-is-polyglot-persistence/]

Different Data Stores are suitable with different requirements and use cases

[Src: https://www.jamesserra.com/archive/2015/07/what-is-polyglot-persistence/]

Functionality	Considerations	Database Type
User Sessions	Rapid Access for reads and writes. No need to be durable.	Key-Value
Financial Data	Needs transactional updates. Tabular structure fits data.	RDBMS
POS Data	Depending on size and rate of ingest. Lots of writes, infrequent reads mostly for analytics.	RDBMS (if modest), Key Value or Document (if ingest very high) or Column if analytics is key.
Shopping Cart	High availability across multiple locations. Can merge inconsistent writes.	Document, (Key Value maybe)
Recommendations	Rapidly traverse links between friends, product purchases, and ratings.	Graph, (Column if simple)
Product Catalog	Lots of reads, infrequent writes. Products make natural aggregates.	Document
Reporting	SQL interfaces well with reporting tools	RDBMS, Column
Analytics	Large scale analytics on large cluster	Column
User activity logs, CSR logs, Social Media analysis	High volume of writes on multiple nodes	Key Value or Document



Example: Multiple AWS services

[src: https://www.allthingsdistributed.com/2018/06/purpose-built-databases-in-aws.html]

Summary: Different Database and Data Modeling Technologies

Databases are built for a purpose and matching the use case with the database will enable developers to write high-performance, scalable, and more functional applications faster.

Developers also are no longer using a single database for all use cases in an application—they are using many databases.

Data is just a starting point…

Why is DATA IMPORTANT?

Data-driven Decision Making and Planning

Data-driven Business Operation

Data-driven Business Strategic Development

Characteristics of Data Quality

Characteristics of Data Quality

Characteristic	How to measure
Accuracy	Is the information correct in every detail?
Completeness	How comprehensive is the information?
Reliability	Does the information contradict other trusted resources?
Relevance	Do you really need this information?
Timeliness	How up- to-date is information? Can it be used for real-time reporting?


Data Analytics Pipeline

Data Analytics Pipeline

Source: https://www.freecodecamp.org/news/scalable-data-analytics-pipeline/

Applications and Business Use Cases

SQL Vs No SQL: What's the different?

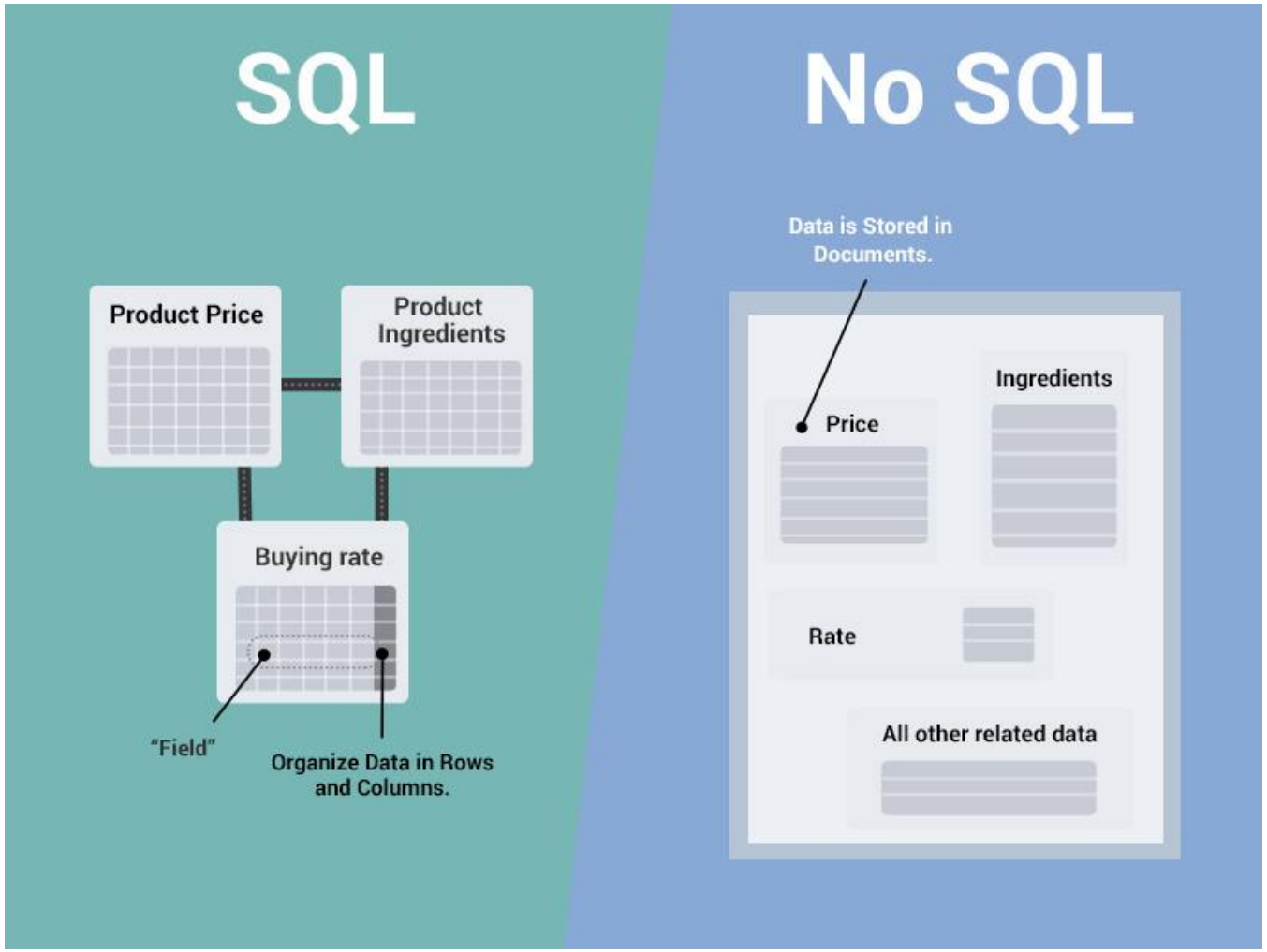


Image src:https://www.apptunix.com/blog/sql-or-nosql-database/

NoSQL: Use Cases

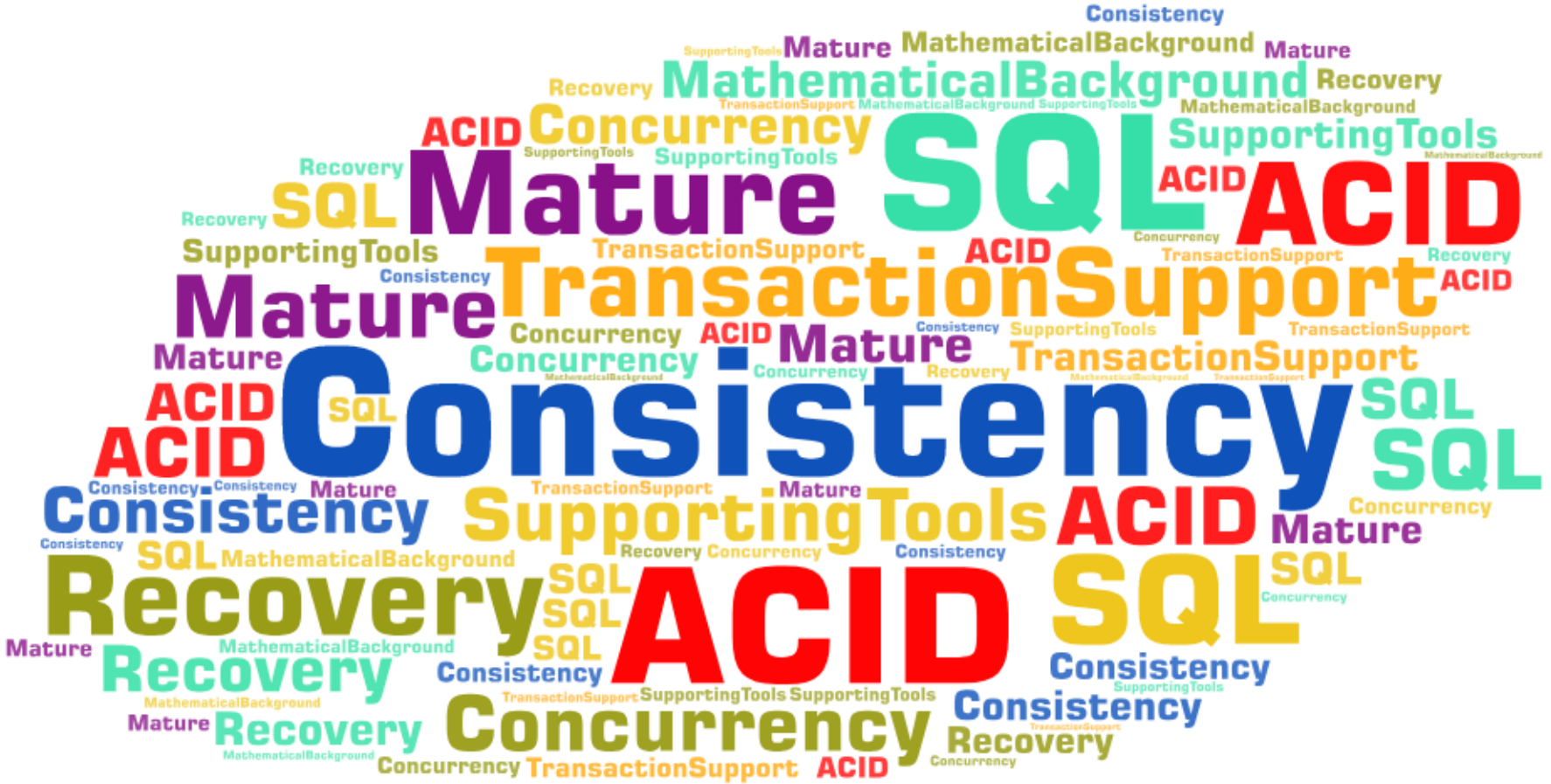
Key Value	Document	Columnar	Graph
Session Management User Preference Shopping Cart	Content Management Web Analytics **Product Catalog** Sigle View E-Commerce	Event Logging Content Management Counters	Social Network Recommendation Social Graph

Recall: Relational Database Concepts & SQL

Menti.com

Relational Database Recap!

CHARACTERISTICS, BENEFITS AND LIMITATIONS



Relational Databases: Characteristics and Benefits

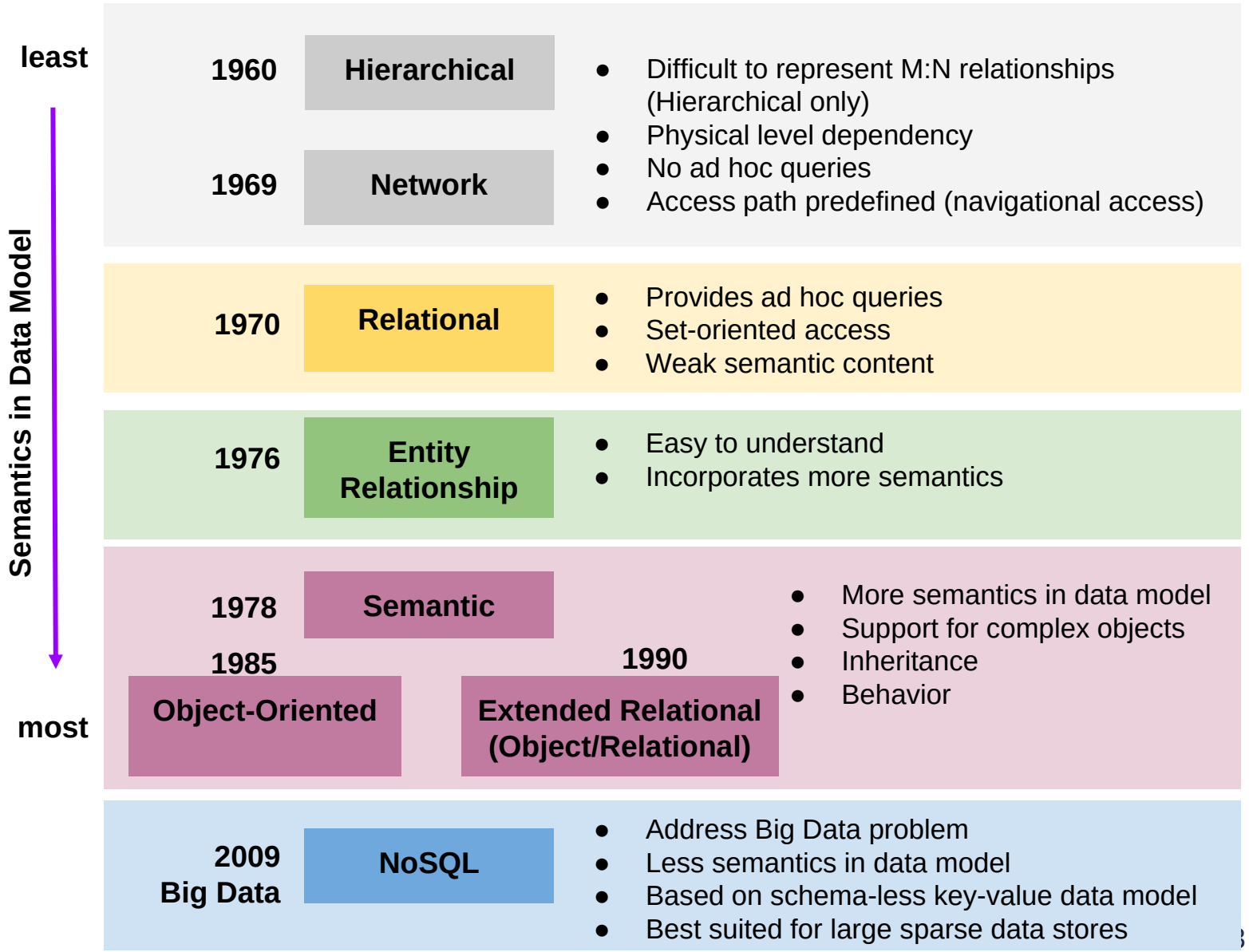
Relational Databases: Limitations

Scalability Issues

- Scale up vs. Scale out (vertical vs. horizontal)
- Not designed to run on clusters / distributed applications
- Joins are expensive

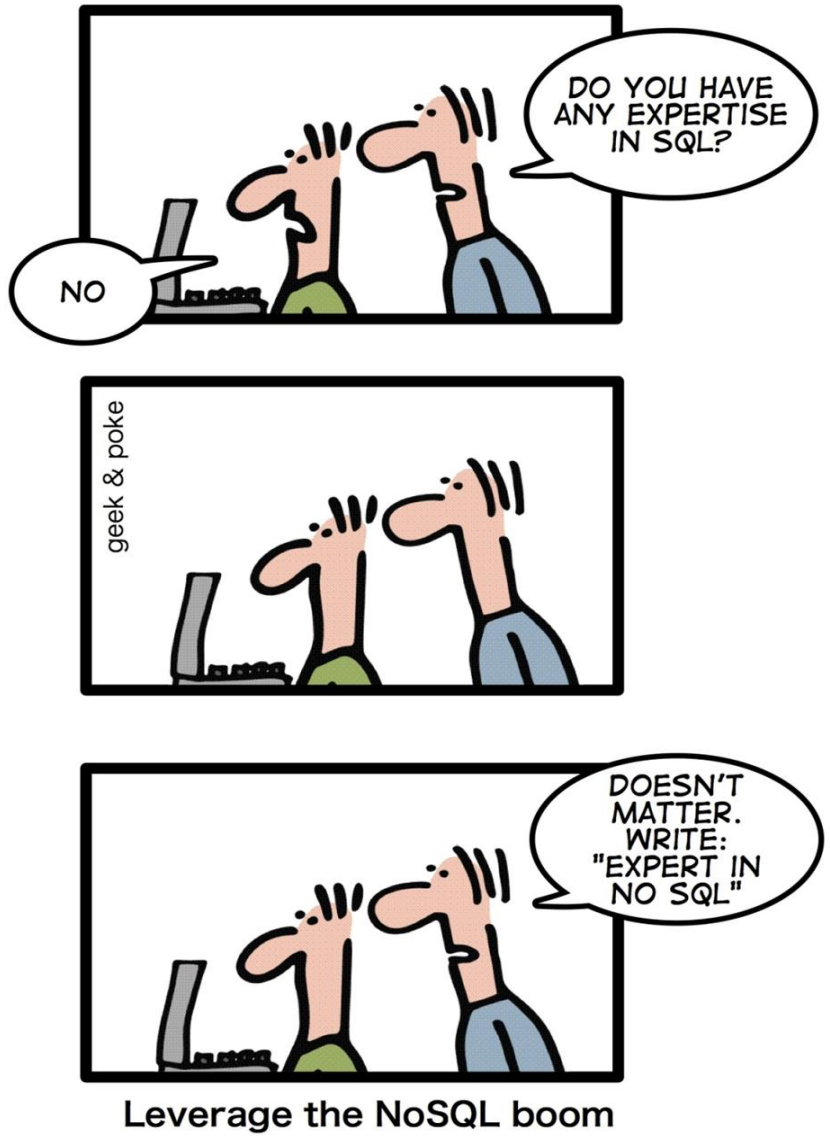
Schema-ful Databases vs. Schema-less Databases

Evolution of data models



NoSQL Database Concepts

HOW TO WRITE A CV



Leverage the NoSQL boom

A Little Humor...

3 DATABASE ADMINS

WALKED INTO

A NOSQL BAR …

A LITTLE WHILE LATER

THEY WALKED OUT BECAUSE

THEY COULDN’T FIND A TABLE

NO SQL

CONCEPTS AND CHARACTERISTICS

NoSQL Origin

Generally newer databases solving new and different problems;

Not only SQL;

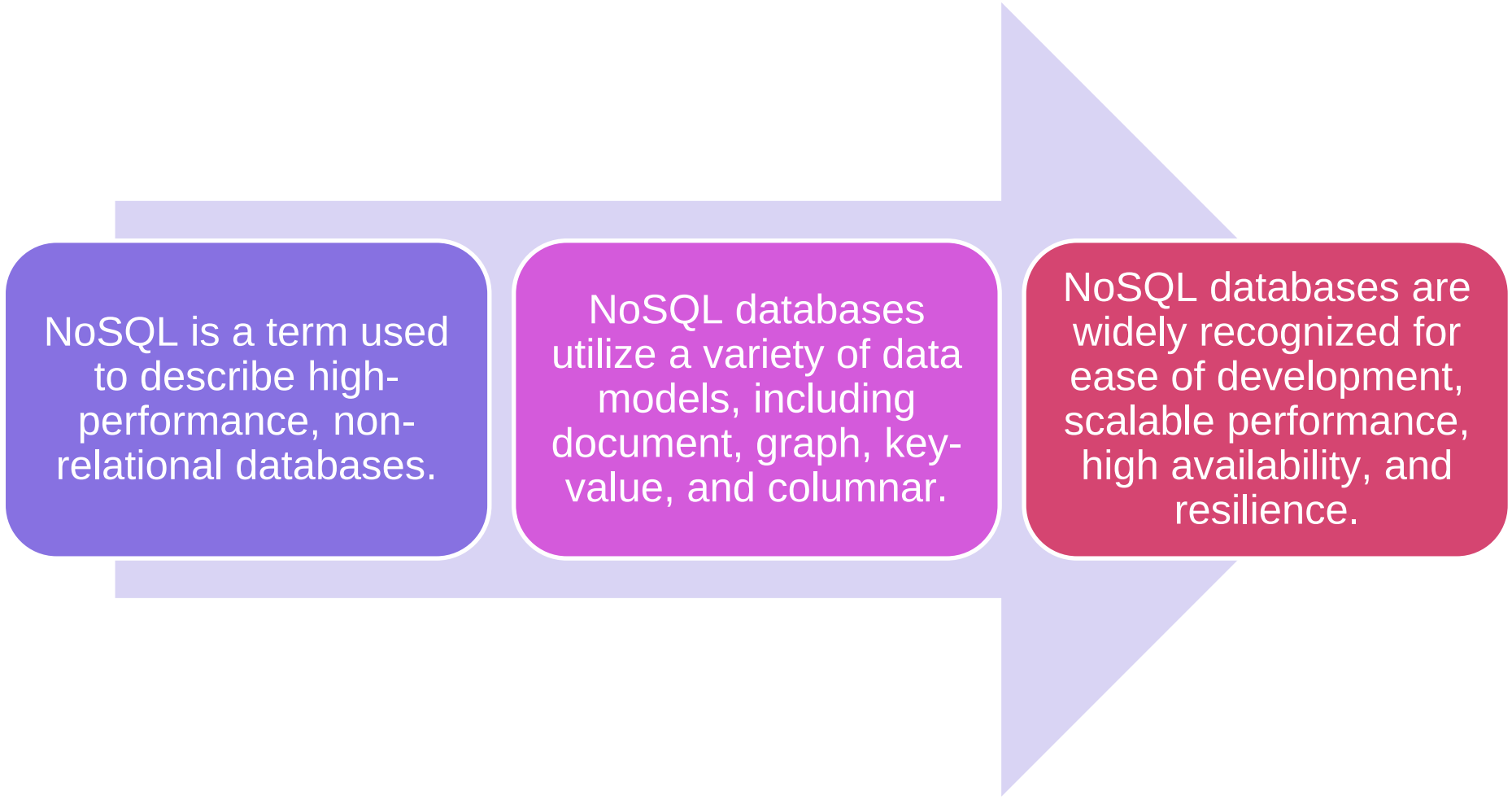
Problems not solved by RDBMSs;

Limitation of RDBMSs, not SQL;

NO SQL

NoSQL is a database technology designed to support the requirements of cloud applications and architected to overcome the scale, performance, data model, and data distribution limitations of relational databases (RDBMS’s).

What is NoSQL?



Schema-less Database: what is?

In Relational DB (schema-ful DB), there are limitations:

- Cannot add a record which does not fit a schema
- Needs to add NULL values to unused data attribute in a record
- Strong datatyping
- Composite attributes and multivalued attributes are not allowed!!

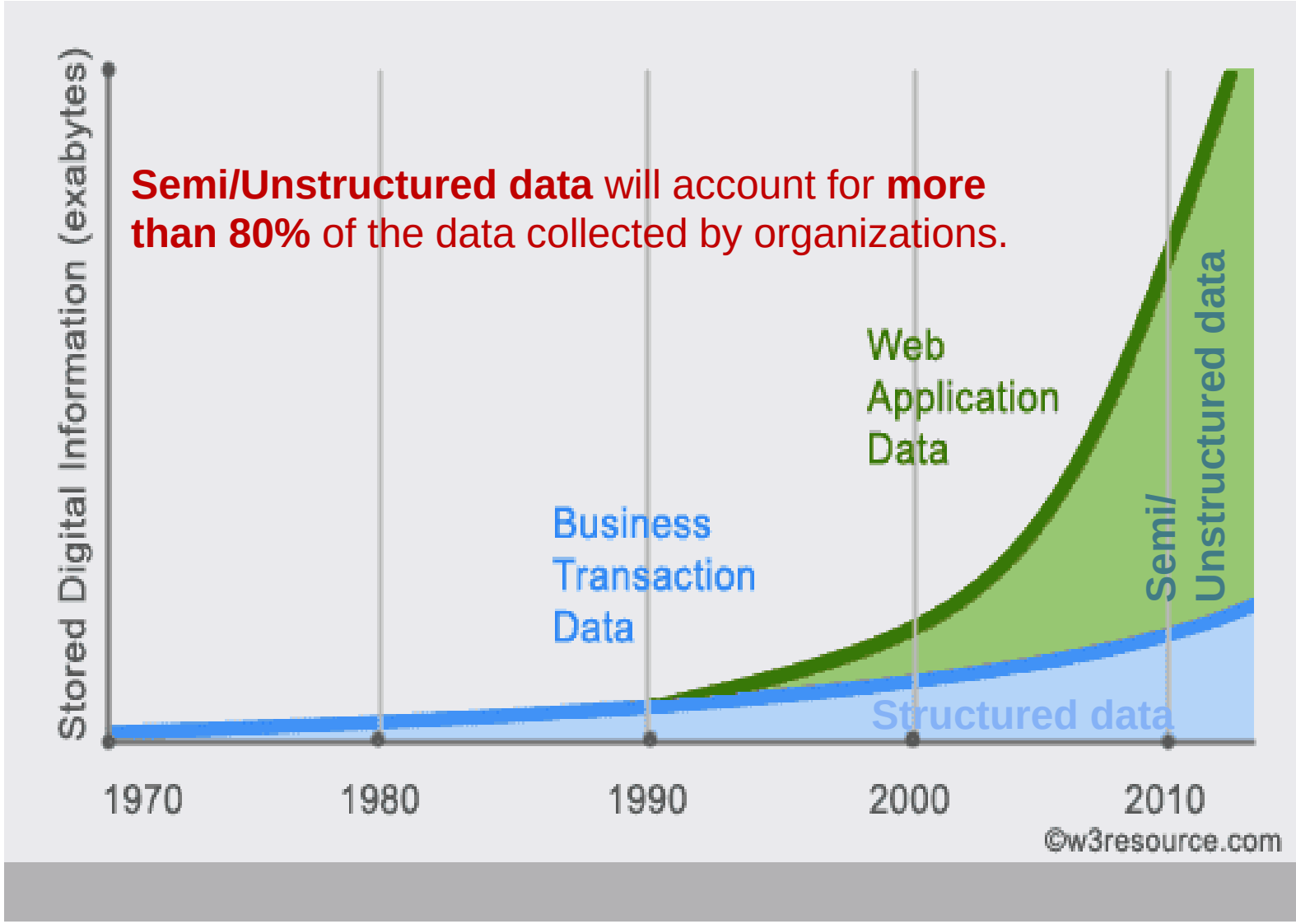
Schema-less Database: what is?

In Schema-less DB

- No fixed, rigid Schema
- No NULL constraint/enforcement
- No datatyping

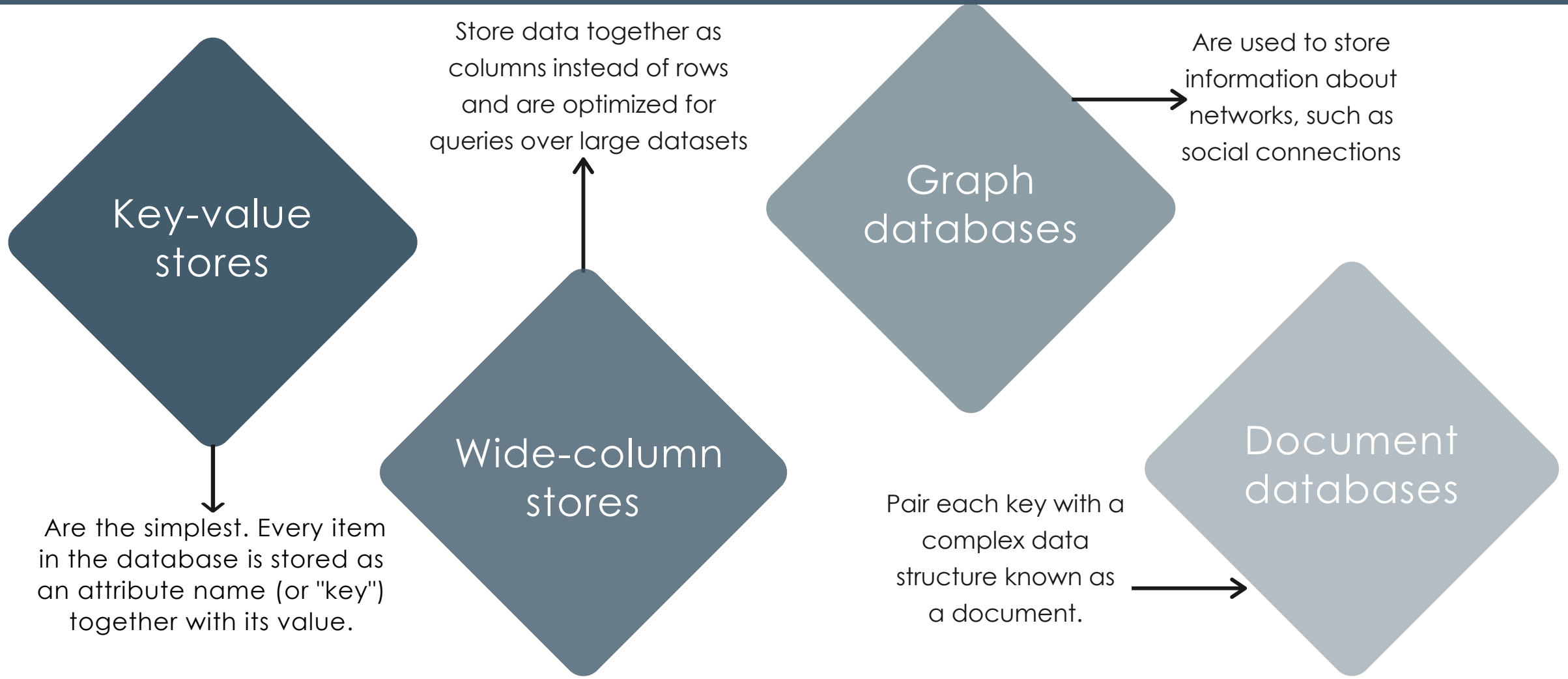
This is Schema-less Database!

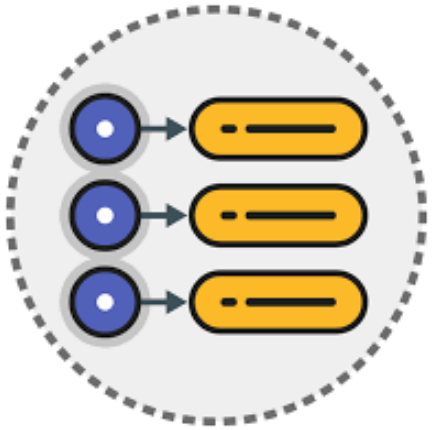
Web Apps Driving Data Growth



Major Categories of NOSQL Data Models

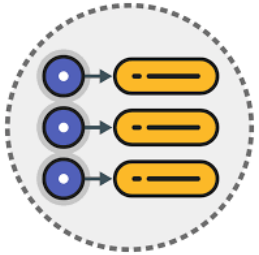
MAJOR CATEGORIES OF NOSQL DATA MODELS





Key-Value Model

Key-Value Model



Key

Value

Nadech

Name: Nadech

Country: Thailand

Age: 30 years

Yaya

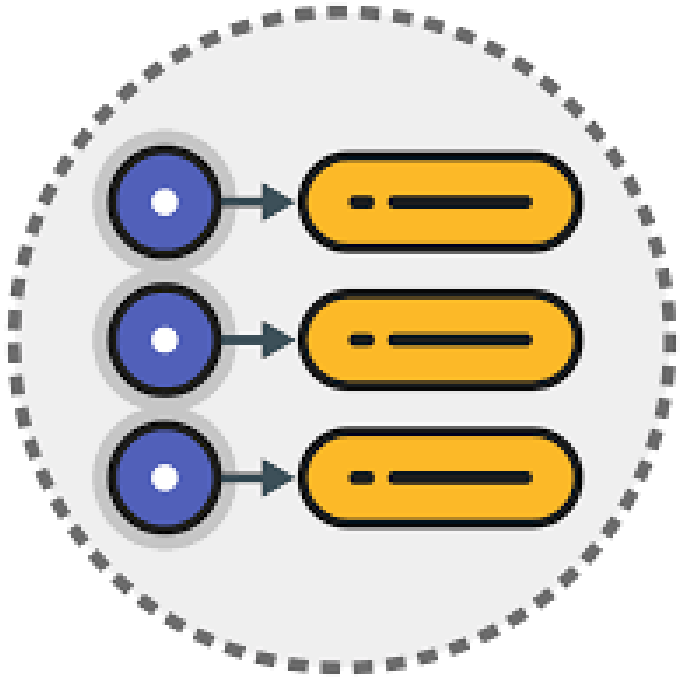
Name: Yaya

Program: IM

Facebook: URL

Video Object

Key-Value Model



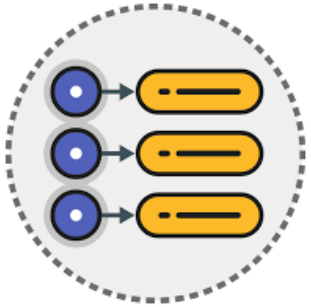
The simplest model: just Keys and Values

- No Schema
- Keys: synthetic or auto-generated
- Values: any object type (e.g., String, JSON, BLOB) stored as uninterpreted block, thus the keys are the only way to retrieve stored data.

Query operations for stored objects are associated with a key:

- PUT, GET, DELETE

Benefits vs. Limitations



BENEFITS

Extremely fast retrieval using the key

Virtually no restriction on the type of data that can be stored:

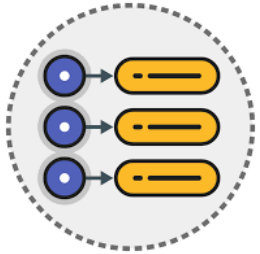
- Text (for example, the HTML code for a Web page)
- Any type of multimedia binary (still images, audio, and video).

LIMITATIONS

Cannot search within stored values rather than always retrieving by the key

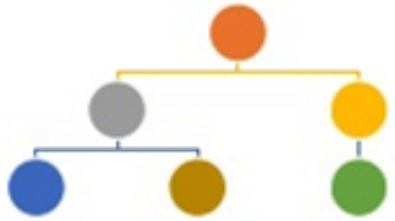
Cannot update parts of a “value” while it’s in the database. You must replace the entire value with a new copy if modifications are needed.

Applications & Use Cases



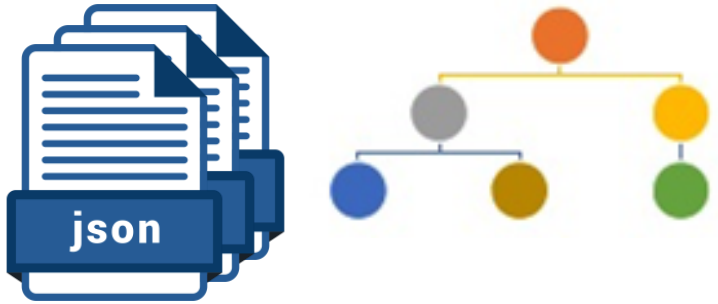
Best suited for applications where access is only through the key.

They are being used for Web sites that include thousands of pages, large image databases, and large catalogs. They are also particularly useful for keeping Web app session information.



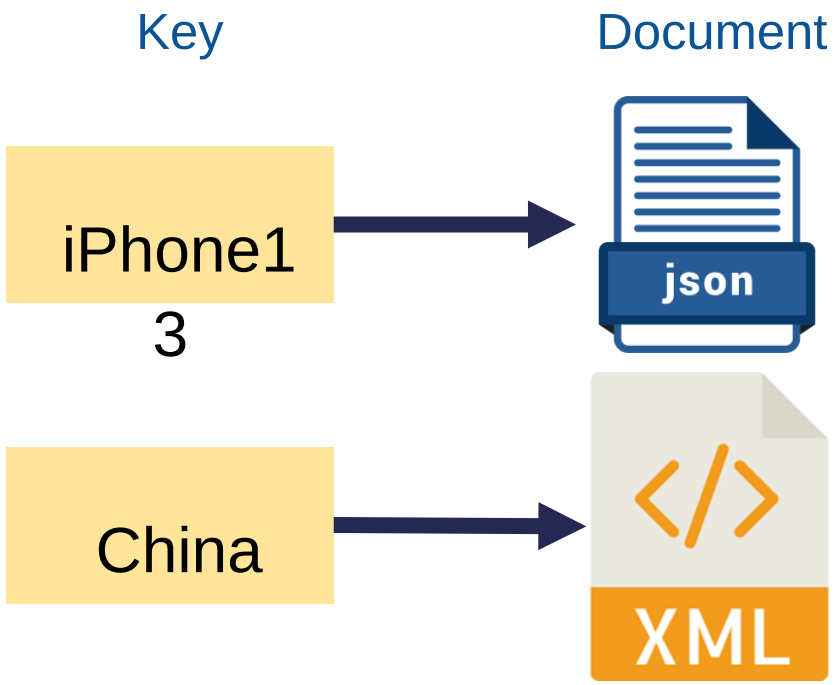
Document Model

Document Model

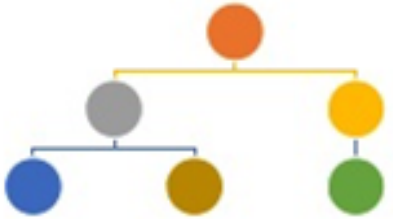


A specialized Key-value Store but rather than storing “values,” it stores “documents”, which are not adhered to schema restrictions.

Provides a way to query the documents based on the contents or metadata.



Document Model



A specialized Key-value Store

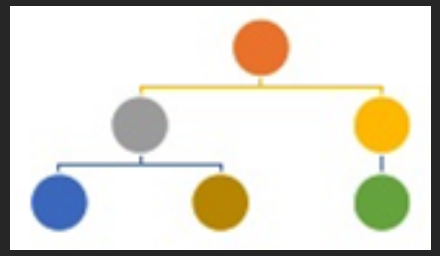
Designed for storing, retrieving and managing document-oriented information, also known as semi-structured data, such as XML, JSON, BSON

Provides APIs or a query/update language that exposes the ability to query or update based on the internal structure in the document.

```
{
   "FirstName": "Bob",
   "Address": "5 Oak St.",
   "Hobby": "sailing"
}
```

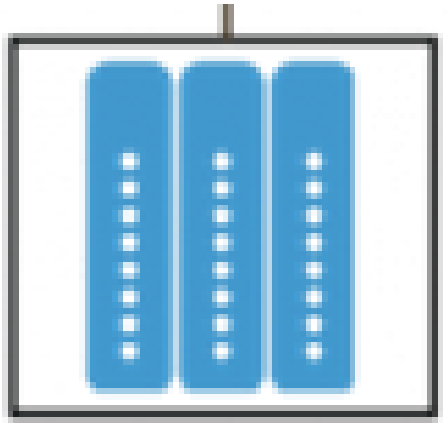

```
<contact>
   <firstname>Bob</firstname>
   <lastname>Smith</lastname>
   <phone type="Cell">(123) 555-0178</phone>
   <phone type="Work">(890) 555-0133</phone>
   <address>
    <type>Home</type>
    <street1>123 Back St.</street1>
    <city>Boys</city>
    <state>AR</state>
    <zip>32225</zip>
    <country>US</country>
   </address>
  </contact>
```

CRUD Operations



- Creation (or insertion)
- Retrieval (or query, search, read or find)
- Update (or edit)
- Deletion (or removal)

Column-Family Model



(AKA. COLUMNAR AND WIDE-COLUMN MODEL)

NOTE: MOST TERMINOLOGY USED HERE ARE BASED ON APACHE CASSANDRA SINCE IT IS ONE OF THE MOST POPULAR COLUMN-FAMILY STORES.

Column-Family Model

Column-family stores **enhance the key-value concept** by providing additional structure.

One of the most influential NoSQL database was Google's BigTable.

Other stores: Cassandra, HBase, Hypertable, Amazon DynamoDB.

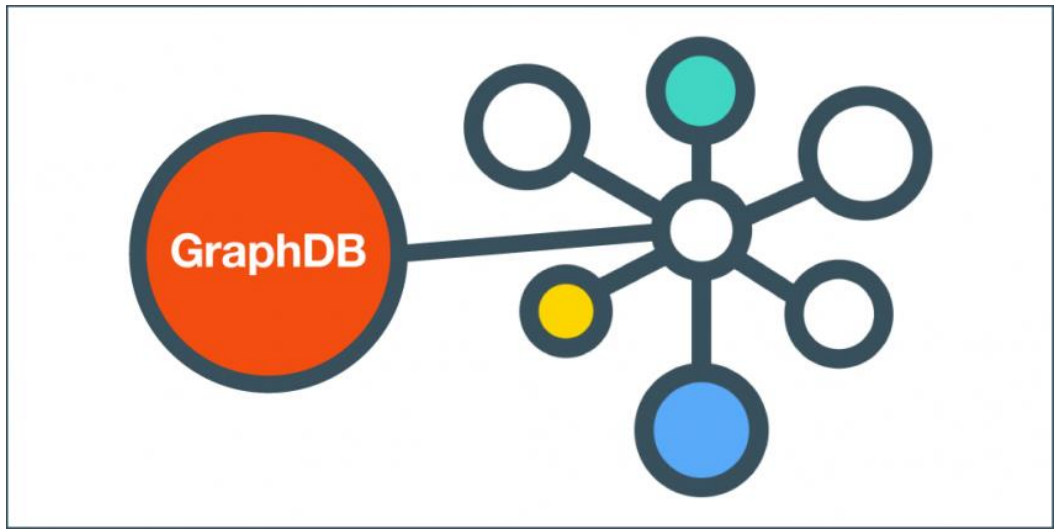
Column-Family Model

Most RDB databases has rows as unit of storage, which helps in writing performances.

In practical use, it has shown to be more efficient for optimizing read operations to store the data in relational tables not per row, but per column.

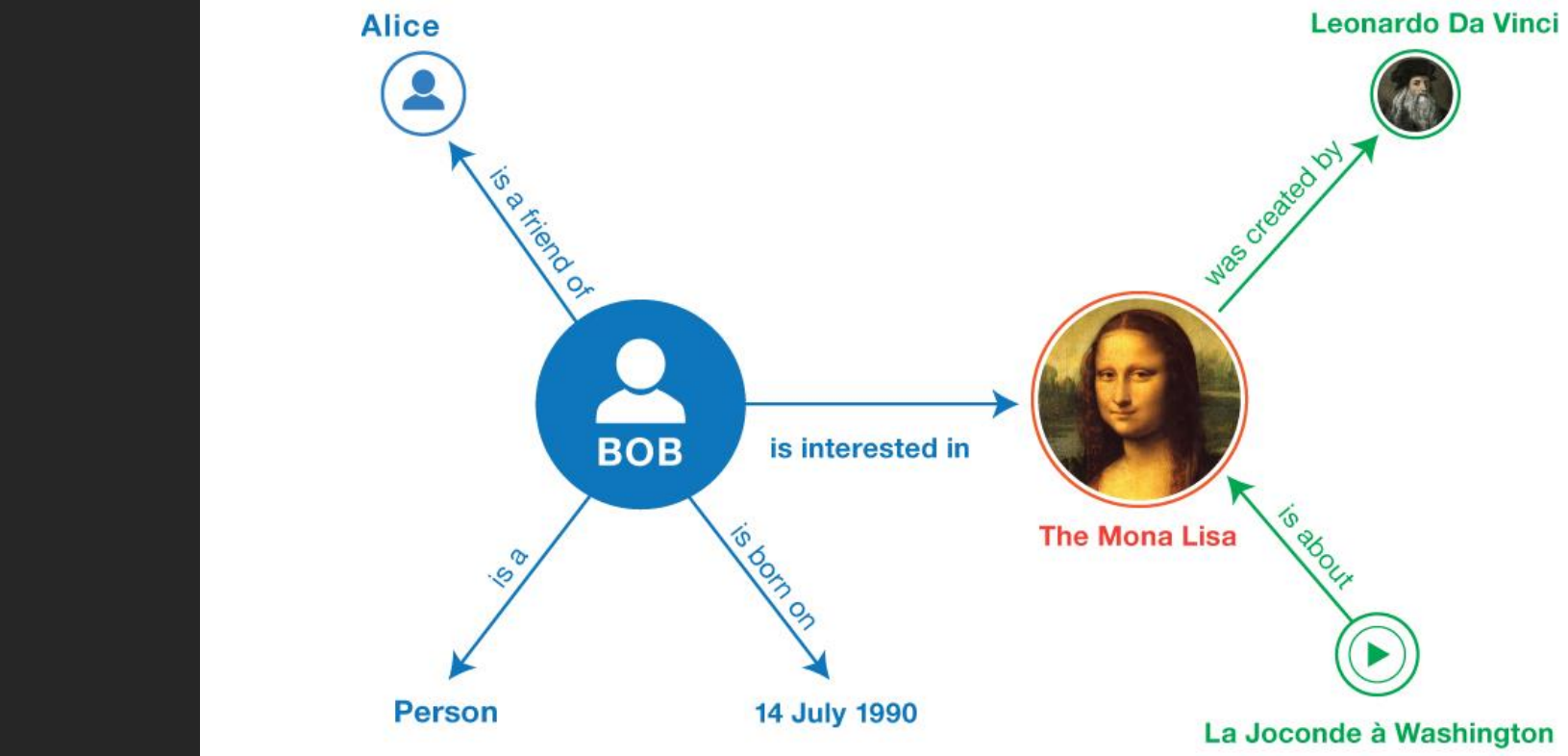
This is because all columns in one row are rarely needed at once, but there are groups of columns that are often read together.

Therefore, in order to optimize access, it is useful to structure the data in such groups of columns—column families—as storage units.



Graph Model

Graph Model (nodes-links-properties structure)



Graph Model

Graph store uses graph structures for semantic queries with nodes, edges and properties to represent and store data.

The relationships allow data in the store to be linked together directly, and in many cases retrieved with one operation.

A query on a graph is known as traversing the graph.

The biggest advantage of the graph store is that joins are not necessary.


THANK YOU