Database Systems Evolution

UA.DETI.CBD

José Luis Oliveira / Carlos Costa

Outline

- Why do we need storage system
- How they evolved along the time
- Milestone solutions
- Current landscape

Thinking about Data Systems

- Many applications today are **data-intensive**, as opposed to **compute-intensive**.
- Raw CPU power is rarely a limiting factor for these applications
  – bigger problems are usually the **amount** of data, the **complexity** of data, and the **speed** at which it is changing.

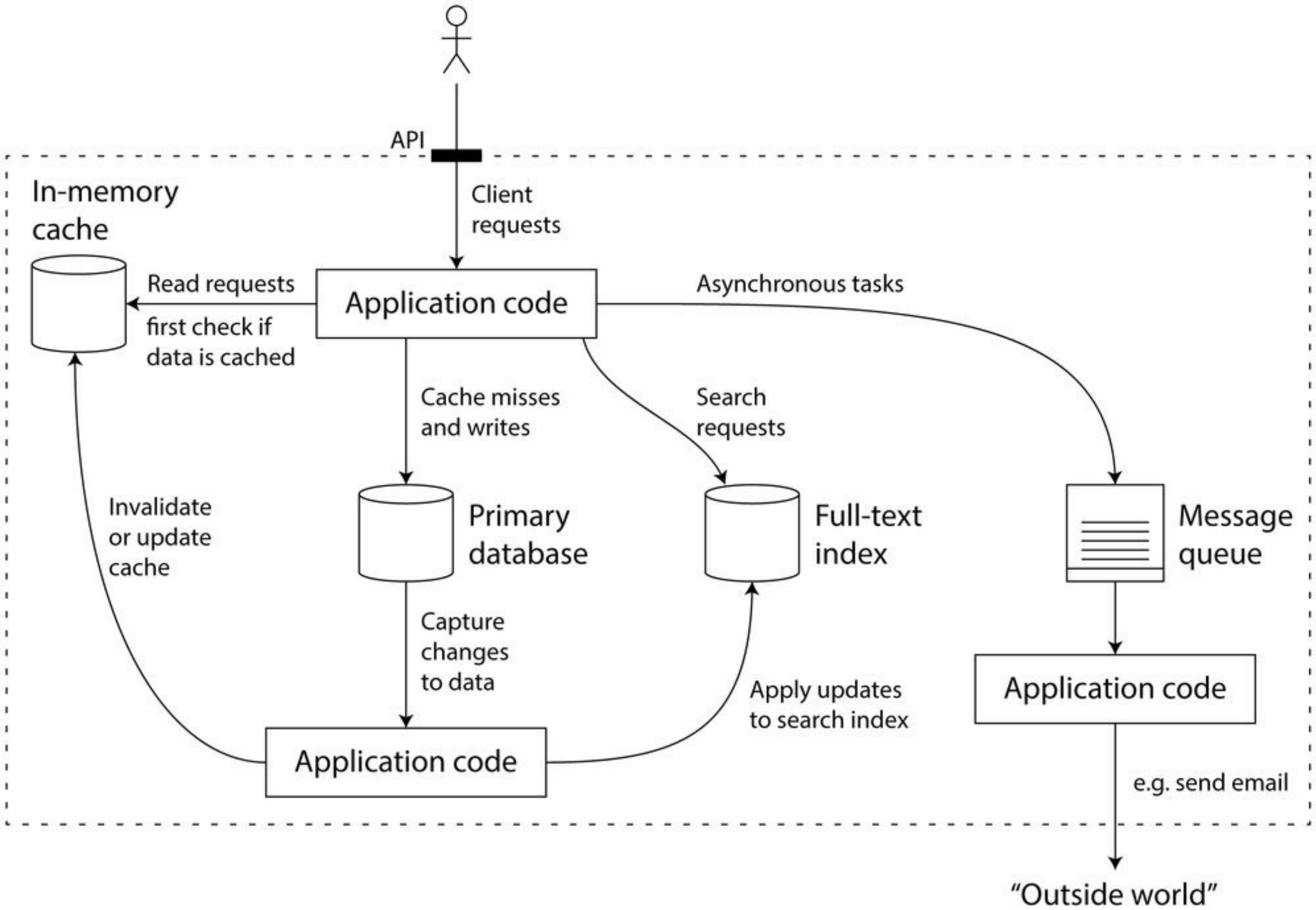
Data systems typically needs to

- Store data so that they, or another application, can find it again later (**databases**).
- Remember the result of an expensive operation, to speed up reads (**caches**).
- Allow users to search data by keyword or filter it in various ways (**search indexes**).
- Send a message to another process, to be handled asynchronously (**message queues**).
- Observe what is happening, and act on events as they occur (**stream processing**).
- Periodically crunch a large amount of accumulated data (**batch processing**).

Thinking about Data Systems

- Increasingly, many applications have wide-ranging requirements
  - Many times, a single tool can no longer meet all of its data processing and storage needs.
- Instead, the work is broken down into tasks that can be performed efficiently on a single tool,
  - the different tools are stitched together using application code.
- For example, we may have an application with:
  - a caching layer (e.g. memcached or similar),
  - a full-text search server (e.g. Elasticsearch or Solr),
  - separated from the main database (e.g. MySQL).

Thinking about Data Systems



Data Systems – some challenges

- How do you ensure that the data remains correct and complete,
  - even when things go wrong internally?
- How do you provide consistently good performance to clients,
  - even when parts of your system are degraded?
- How do you scale to handle an increase in load?
- What does a good API for the service look like?

Data Systems – some requirements

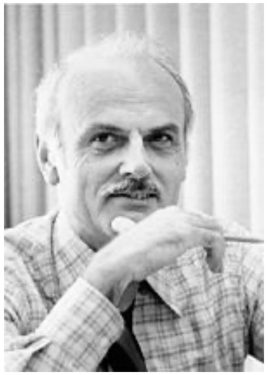
- **Reliability**: The system should continue performing the correct function at the desired performance,
  - even in the face of adversity (hardware or software faults, and even human error).
- **Scalability**: As the system grows (in data volume, traffic volume or complexity), there should be reasonable ways of dealing with that growth.
- **Maintainability**: Over time, many different people should all be able to work on it productively,
  - Engineering and operations, both maintaining current behavior and adapting the system to new use cases.

Database Systems

- A "database" is normally referred as a **set of related data** and its **organization**.
- A "database management system" (**DBMS**) controls the access to this data.
  - Providing functions that allow writing, searching, updating, retrieving, and removing large quantities of information.

Brief History of Database Systems

- Pre-relational era (1970's)
  - Hierarchical (IMS), Network (Codasyl)
  - Many database systems
    - Complex data structures and low-level query language
    - Incompatible, exposing many implementation details
- **Relational DBMSs** (1980s)
  - Edgar F. Codd's relational model in 1970
  - Powerful high-level query language
  - A few major DB systems dominated the market



- Object-Oriented DBMSs (1990s)
  - Motivated by "mismatch" between RDBMS and OO PL
  - Persistent types in C++, Java or Small Talk
  - Issues: Lack of high level QL, no standards, performance

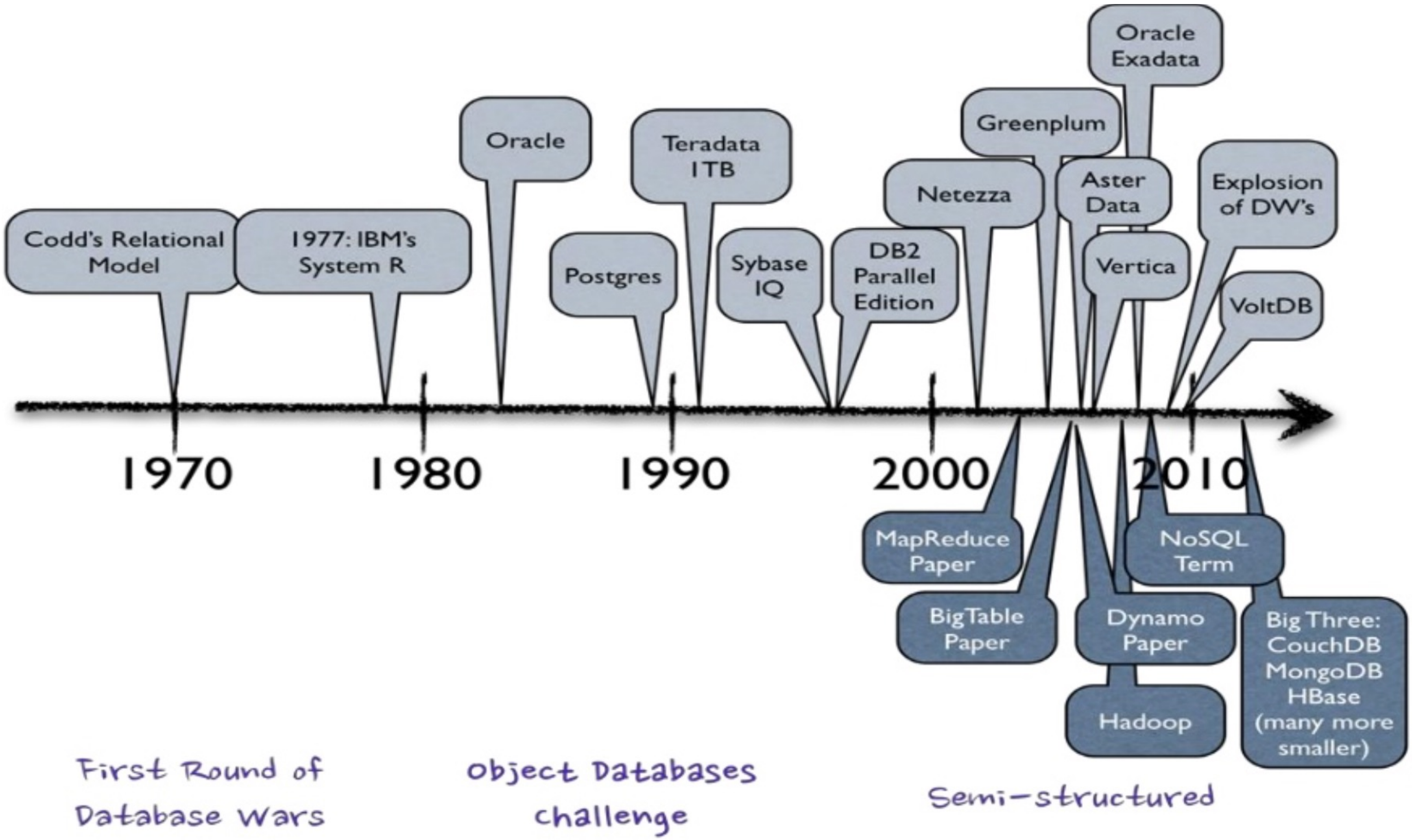
Brief History of Database Systems

- Object-relational DBMS (OR-DBMS) (1990s)
  - Relational DBMS vendors' answer to OO
  - User-defined types, functions (spatial, multimedia) Nested tables
  - SQL: 1999 (2003) standards. Plus performance.
- XML/DBMS (2000s)
  - Web and XML are merging
  - Native support of XML through ORDBMS extension or native XML DBMS
- Data analytics system (DSS) (2000s)
  - **Data warehousing and OLAP**

Brief History of Database Systems

- Data stream management systems (2000s)
  - Continuous query against data streams
- The era of big data (mid 2000-now):
  - **Big data**: datasets that grow so large (terabytes to petabytes) that they become awkward to work with traditional DBMS
  - Parallel DBMSs continue to push the scale of data
  - **MapReduce** dominates on Web data analysis
  - **NoSQL** (not only SQL) is fast growing

Database Evolution Timeline

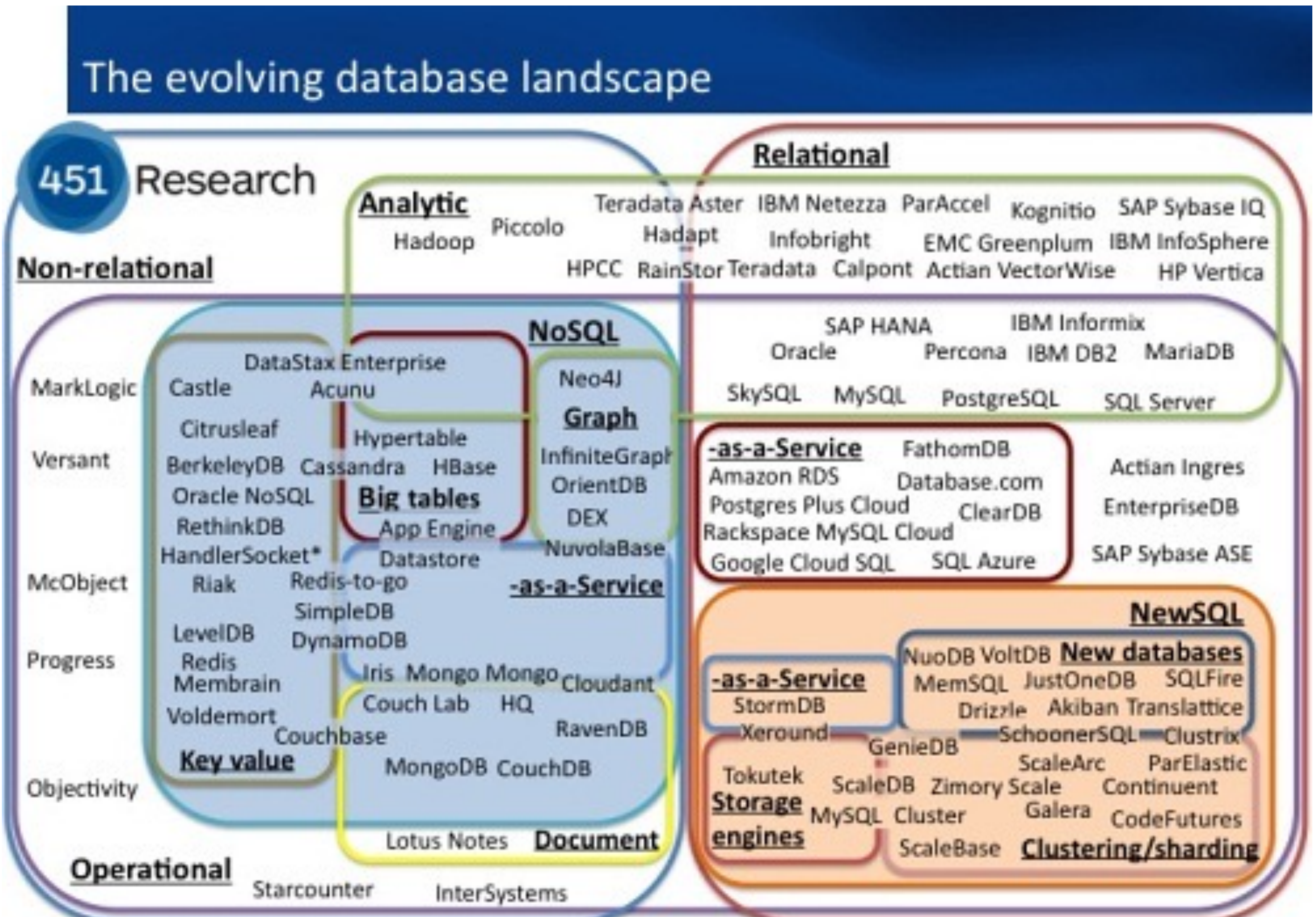


"Big data – an introduction", Peter Morgan

Database Systems Landscape

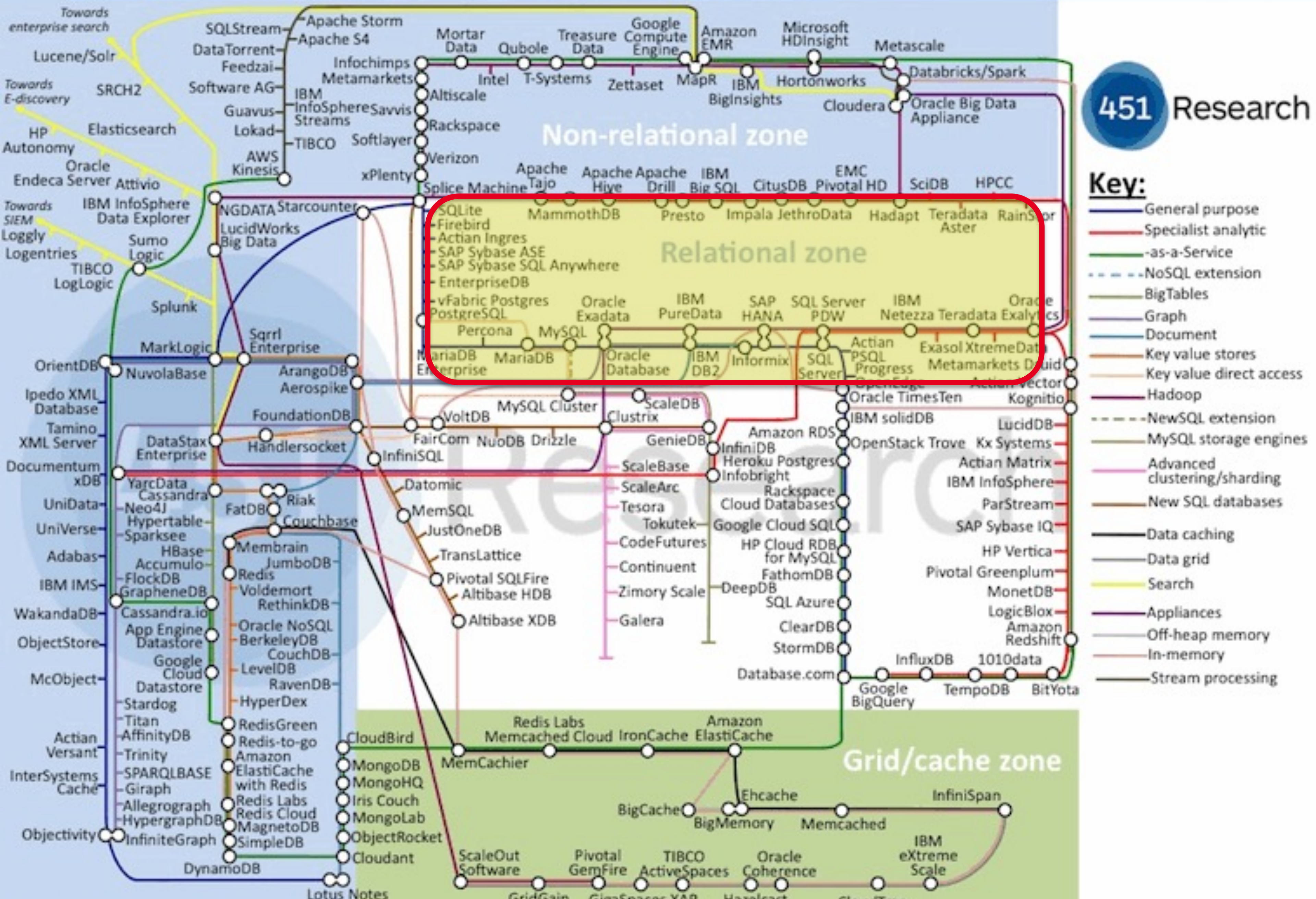
ORACLE
Microsoft SQL Server
SYBASE
PostgreSQL
DB2
TERADATA
amazon DynamoDB
MySQL
Redis
OrientDB
mongoDB
riak
MEMCACHED
Cassandra
HBASE
CouchDB relax
Neo4j

Database Systems Landscape



The 451 Group

Data Platforms Landscape Map – February 2014

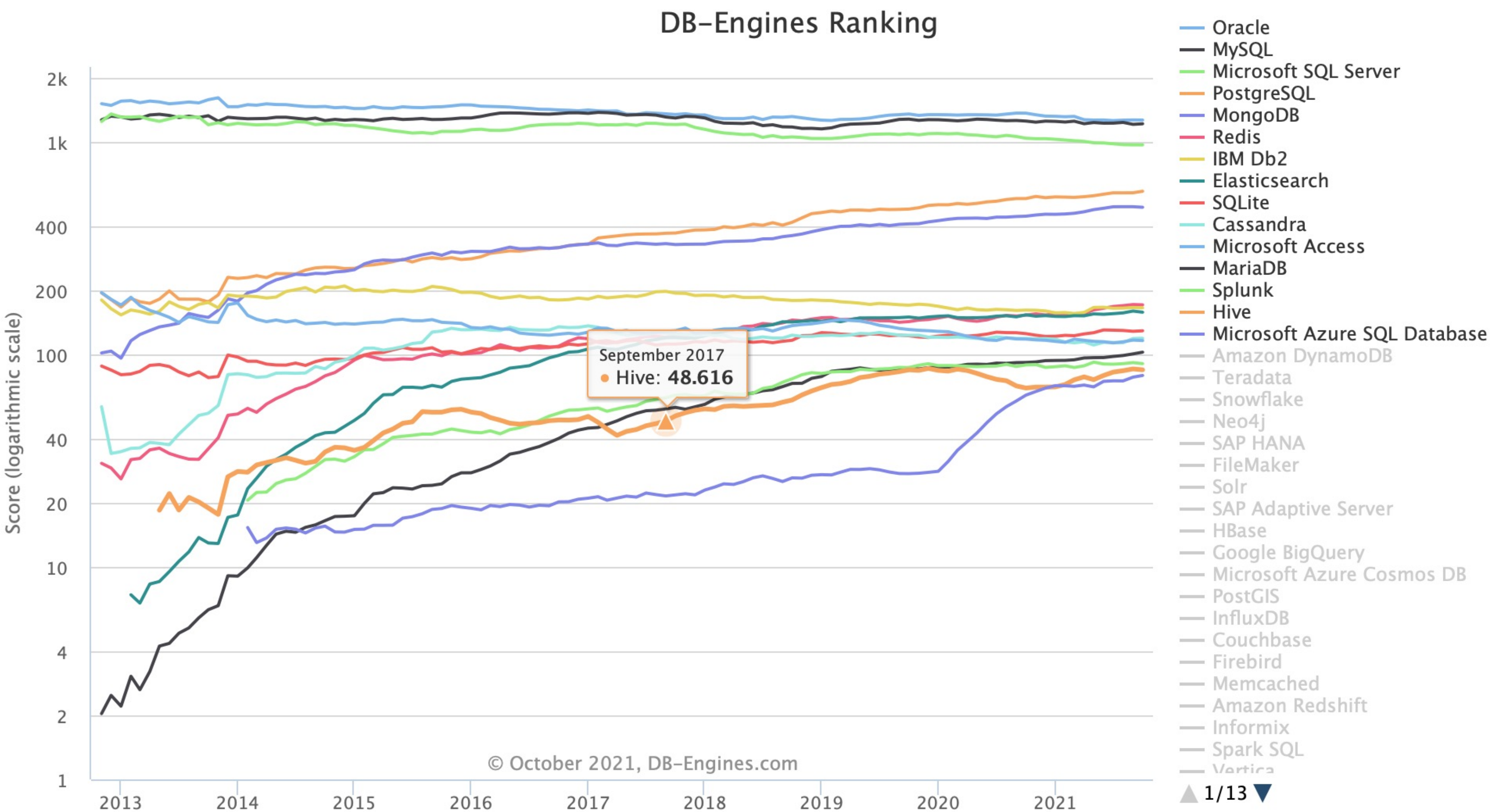


Database Systems Landscape

Rank			DBMS	Database Model	Score		
Oct 2021	Sep 2021	Oct 2020			Oct 2021	Sep 2021	Oct 2020
1.	1.	1.	Oracle	Relational, Multi-model	1270.35	-1.19	-98.42
2.	2.	2.	MySQL	Relational, Multi-model	1219.77	+7.24	-36.61
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	970.61	-0.24	-72.51
4.	4.	4.	PostgreSQL	Relational, Multi-model	586.97	+9.47	+44.57
5.	5.	5.	MongoDB	Document, Multi-model	493.55	-2.95	+45.53
6.	6.	↑ 8.	Redis	Key-value, Multi-model	171.35	-0.59	+18.07
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model	165.96	-0.60	+4.06
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model	158.25	-1.98	+4.41
9.	9.	9.	SQLite	Relational	129.37	+0.72	+3.95
10.	10.	10.	Cassandra	Wide column	119.28	+0.29	+0.18
11.	11.	11.	Microsoft Access	Relational	116.38	-0.56	-1.87
12.	12.	12.	MariaDB	Relational, Multi-model	102.59	+1.90	+10.82
13.	13.	13.	Splunk	Search engine	90.61	-0.99	+1.21
14.	14.	↑ 15.	Hive	Relational	84.74	-0.83	+15.19
15.	15.	↑ 17.	Microsoft Azure SQL Database	Relational, Multi-model	79.72	+1.46	+15.32
16.	16.	16.	Amazon DynamoDB	Multi-model	76.55	-0.38	+8.14
17.	17.	↓ 14.	Teradata	Relational, Multi-model	69.83	+0.15	-5.96
18.	↑ 21.	↑ 64.	Snowflake	Relational	58.26	+6.19	+52.32
19.	↓ 18.	↑ 21.	Neo4j	Graph	57.87	+0.24	+6.53
20.	↓ 19.	↓ 19.	SAP HANA	Relational, Multi-model	55.28	-0.96	+1.04
21.	↓ 20.	↑ 23.	FileMaker	Relational	52.84	+0.52	+5.46
22.	22.	↓ 20.	Solr	Search engine, Multi-model	51.17	+1.36	-1.31

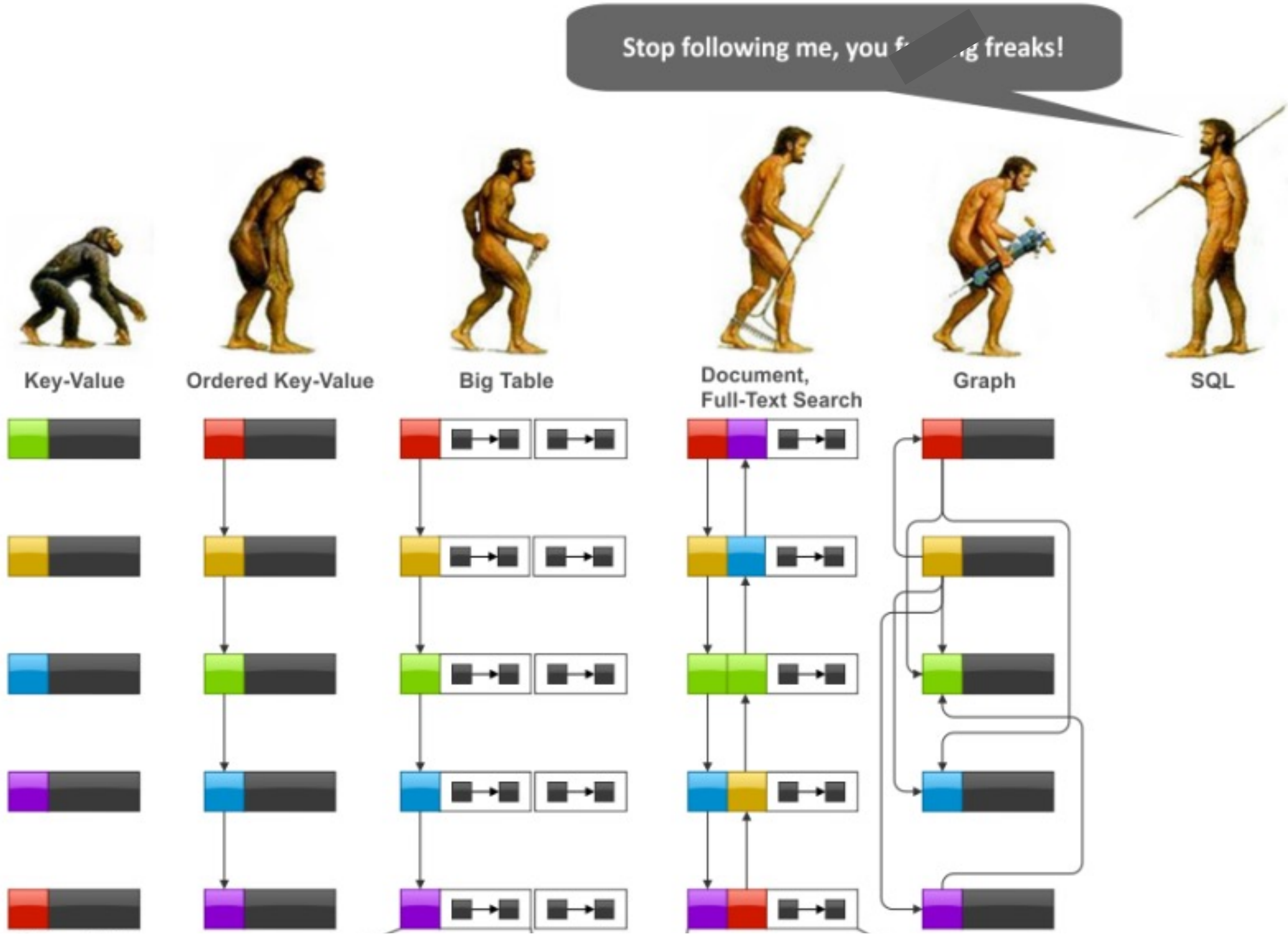
https://db-engines.com/en/ranking

Database Systems Landscape



https://db-engines.com/en/ranking_trend

Database Systems Landscape



https://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/

Resources

- Martin Kleppmann, ***Designing Data-Intensive Applications***, O'Reilly Media, Inc., 2017.
- Pramod J Sadalage and Martin Fowler, ***NoSQL Distilled*** Addison-Wesley, 2012.
- Eric Redmond, Jim R. Wilson. ***Seven databases in seven weeks***, Pragmatic Bookshelf, 2012.
- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, ***Database systems: the complete book*** *(2nd Ed.)*, Pearson Education, 2009.