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.

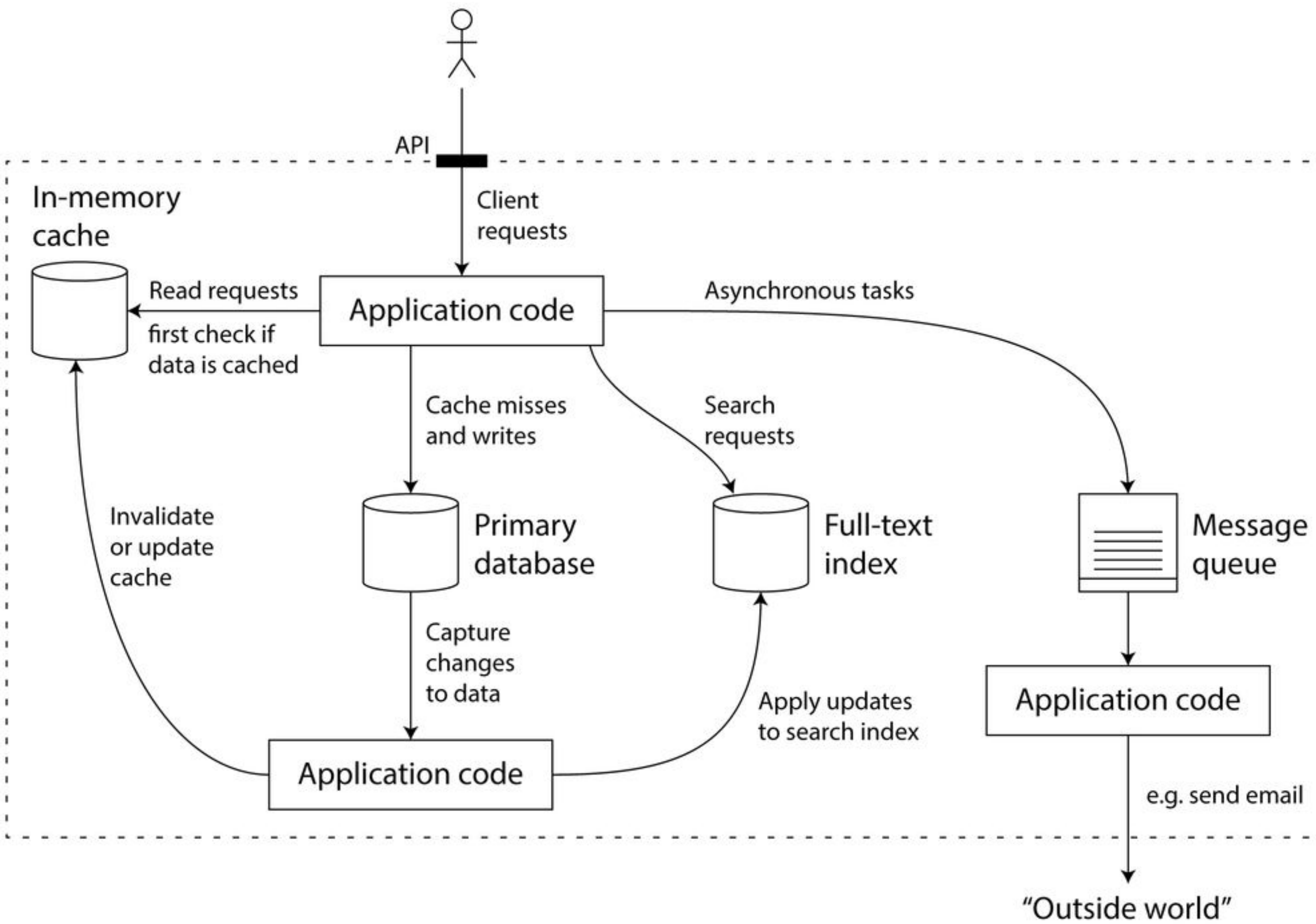
Data systems typically needs to

- 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.

Data Systems

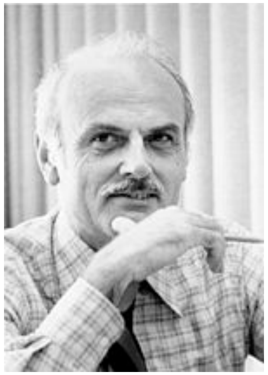
- Some fundamental ways of thinking about data-intensive applications.
  - Reliability means making systems work correctly, even when faults occur.
    - **Fault tolerance** techniques can hide certain types of fault from the end user.
  - Scalability means having strategies for keeping performance good, even when load increases.
    - we need to describe **load and performance** quantitatively.
  - Maintainability it's about making life better for the engineering and operations teams.
    - **Good abstractions** can help reduce complexity and make the system easier to modify and adapt for 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

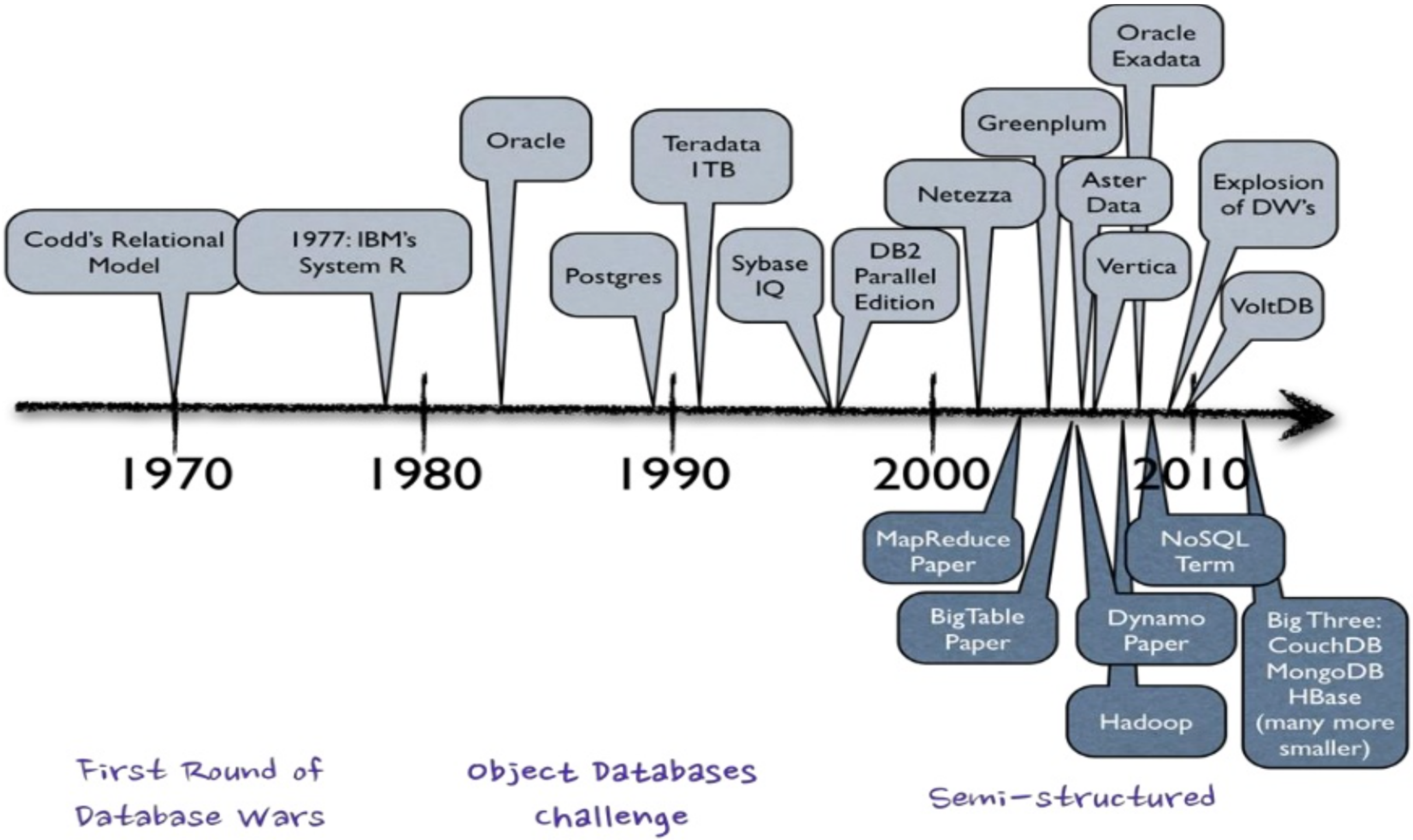
Brief History of Database Systems

- 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

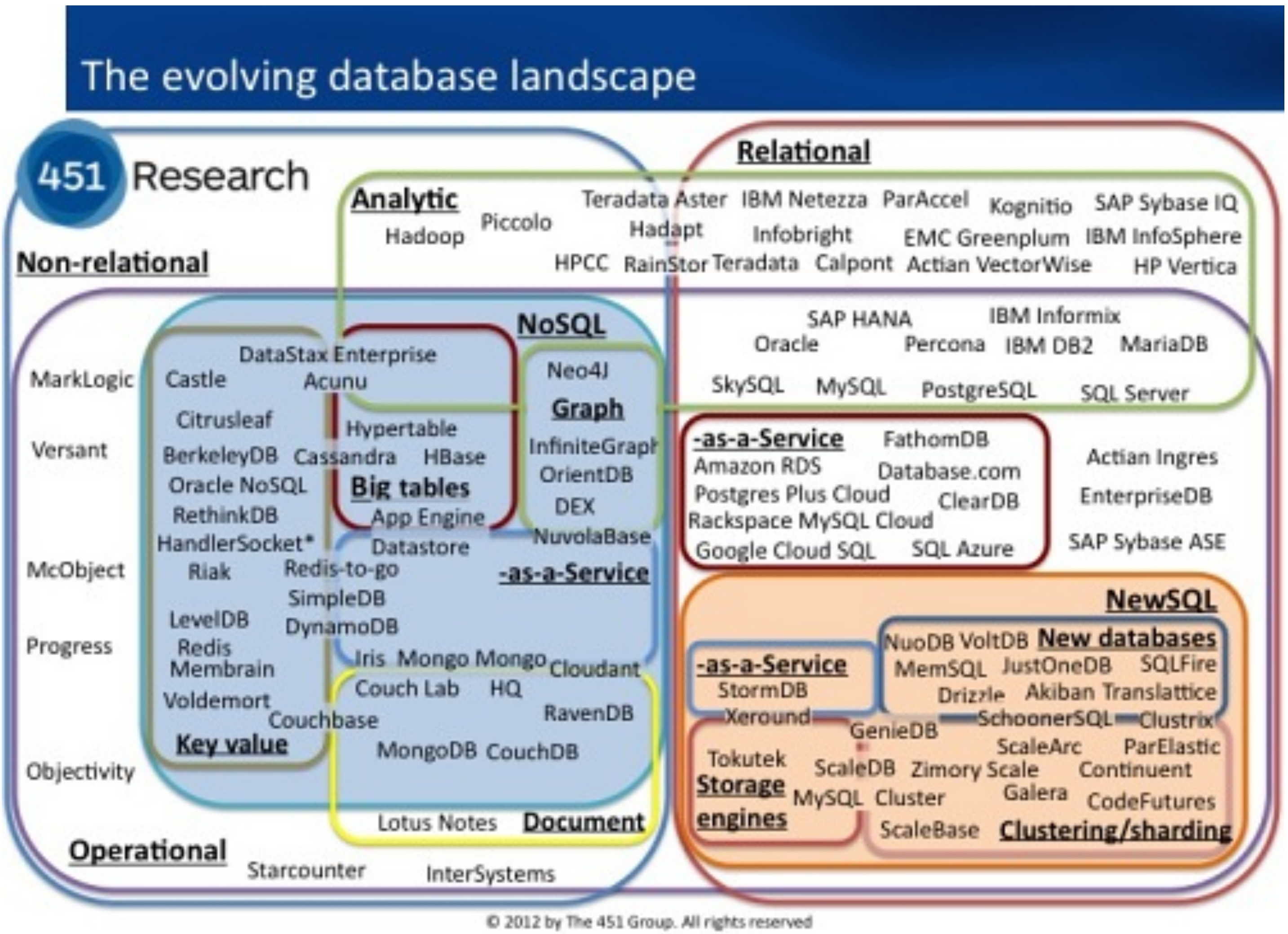


"Big data – an introduction", Peter Morgan

Database Systems Landscape

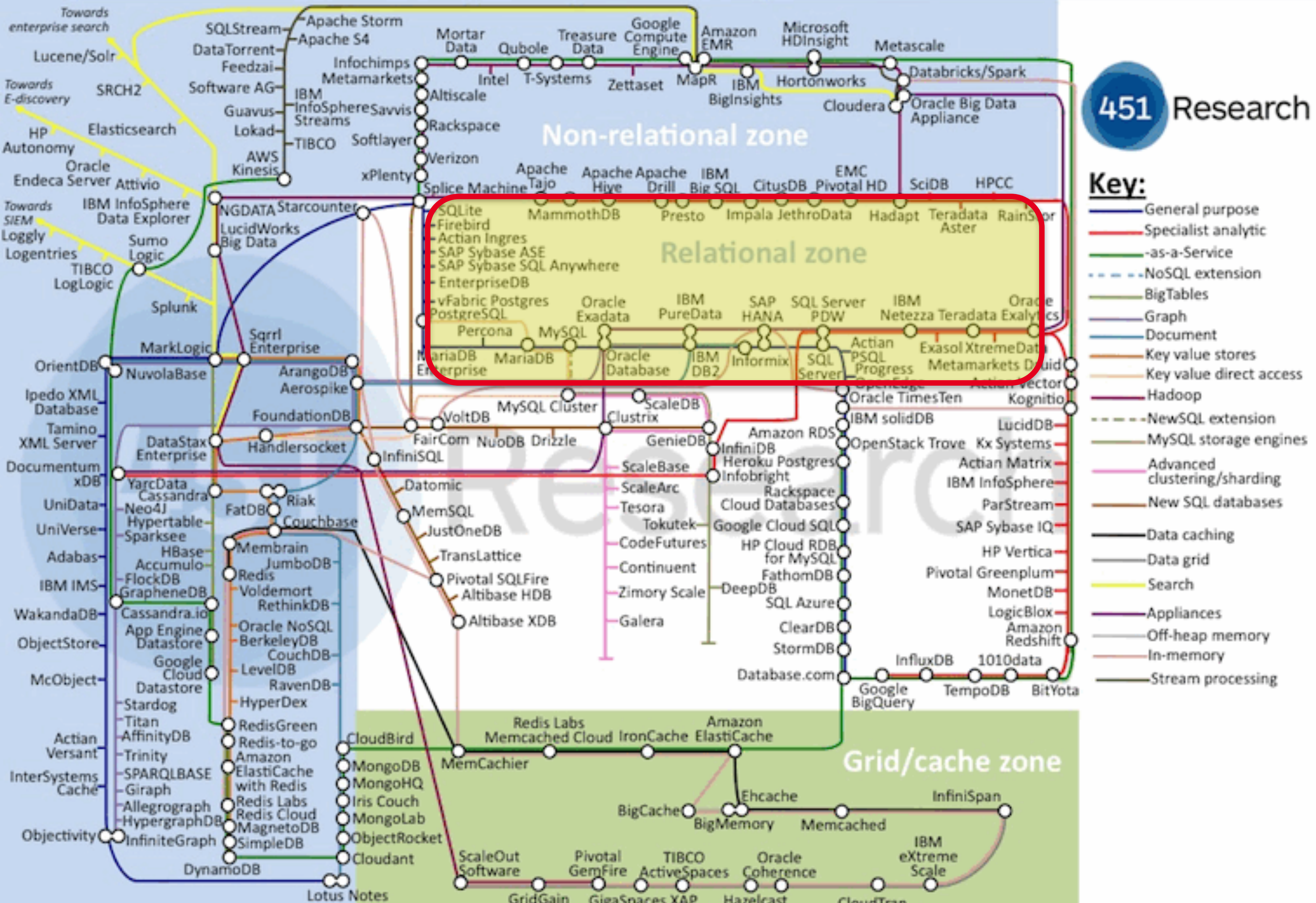
ORACLE
Microsoft SQL Server
SYBASE
PostgreSQL
DB2
TERADATA
amazon DynamoDB
MySQL
Redis
OrientDB
mongoDB
riak
MEMCACHED
Cassandra
HBASE
CouchDB relax
Neo4j

Database Systems Landscape



The 451 Group

Data Platforms Landscape Map – February 2014



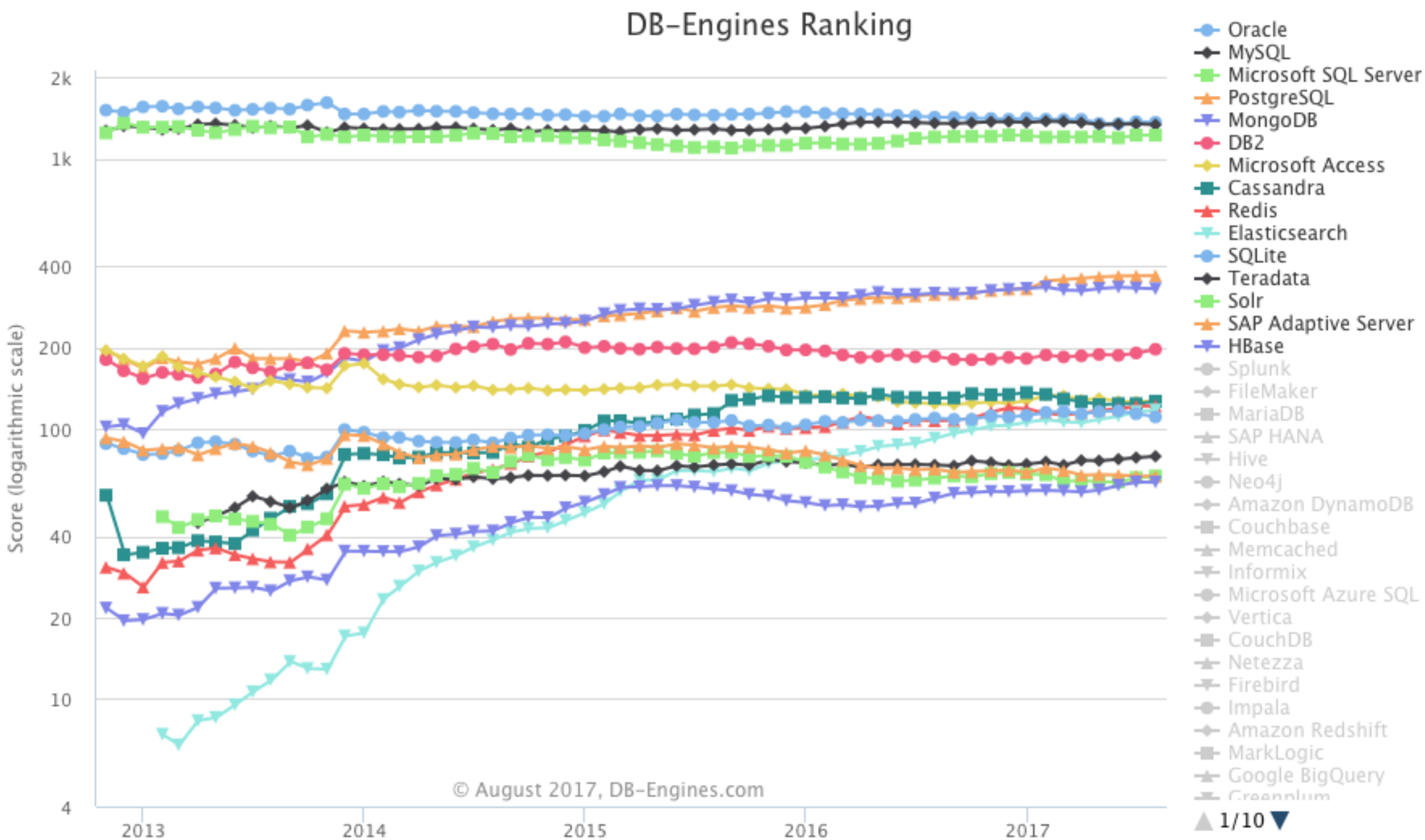
Database Systems Landscape

331 systems in ranking, August 2017

Rank Aug 2017	Rank Jul 2017	Rank Aug 2016	DBMS	Database Model	Score Aug 2017	Score Jul 2017	Score Aug 2016
1.	1.	1.	Oracle	Relational DBMS	1367.88	-7.00	-59.85
2.	2.	2.	MySQL	Relational DBMS	1340.30	-8.81	-16.73
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1225.47	-0.52	+20.43
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	369.76	+0.32	+54.51
5.	5.	↓ 4.	MongoDB	Document store	330.50	-2.27	+12.01
6.	6.	6.	DB2	Relational DBMS	197.47	+6.22	+11.58
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	127.03	+0.90	+2.98
8.	8.	↓ 7.	Cassandra	Wide column store	126.72	+2.60	-3.52
9.	9.	↑ 10.	Redis	Key-value store	121.90	+0.38	+14.57
10.	10.	↑ 11.	Elasticsearch	Search engine	117.65	+1.67	+25.16
11.	11.	↓ 9.	SQLite	Relational DBMS	110.85	-3.02	+0.99
12.	12.	12.	Teradata	Relational DBMS	79.23	+0.86	+5.59
13.	↑ 14.	↑ 14.	Solr	Search engine	66.96	+0.93	+1.18
14.	↓ 13.	↓ 13.	SAP Adaptive Server	Relational DBMS	66.92	+0.00	-4.13
15.	15.	15.	HBase	Wide column store	63.52	-0.10	+8.01
16.	16.	↑ 17.	Splunk	Search engine	61.46	+1.17	+12.56
17.	17.	↓ 16.	FileMaker	Relational DBMS	59.65	+1.00	+4.64
18.	18.	↑ 20.	MariaDB	Relational DBMS	54.70	+0.33	+17.82
19.	19.	19.	SAP HANA	Relational DBMS	47.97	+0.03	+5.24
20.	20.	↓ 18.	Hive	Relational DBMS	47.30	+1.10	-0.51
21.	21.	21.	Neo4j	Graph DBMS	38.00	-0.52	+2.43
22.	22.	↑ 25.	Amazon DynamoDB	Document store	37.62	+1.16	+11.02
23.	23.	↑ 24.	Couchbase	Document store	32.97	-0.05	+5.57

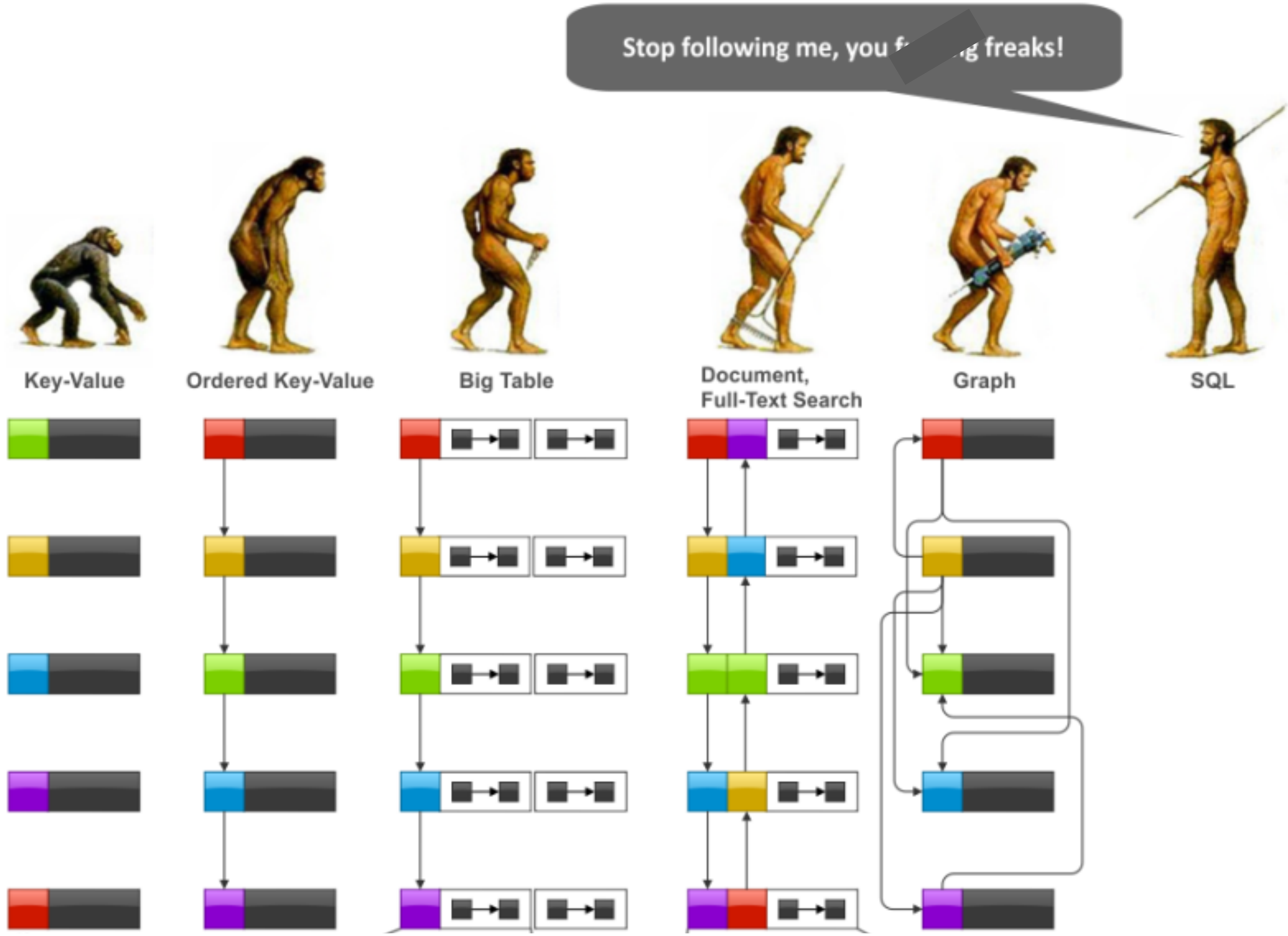
https://db-engines.com/en/ranking

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.