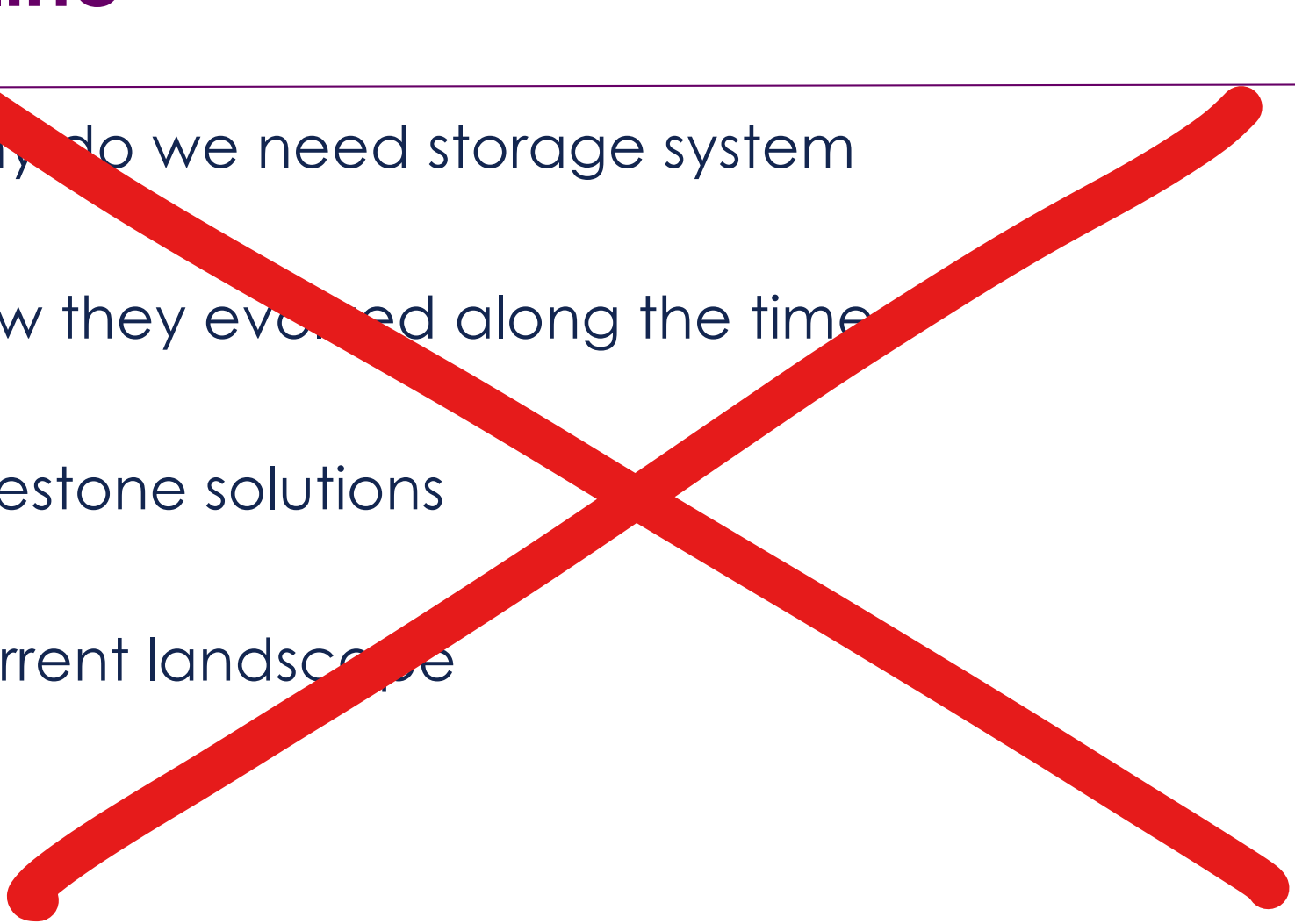
Database Systems Evolution

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

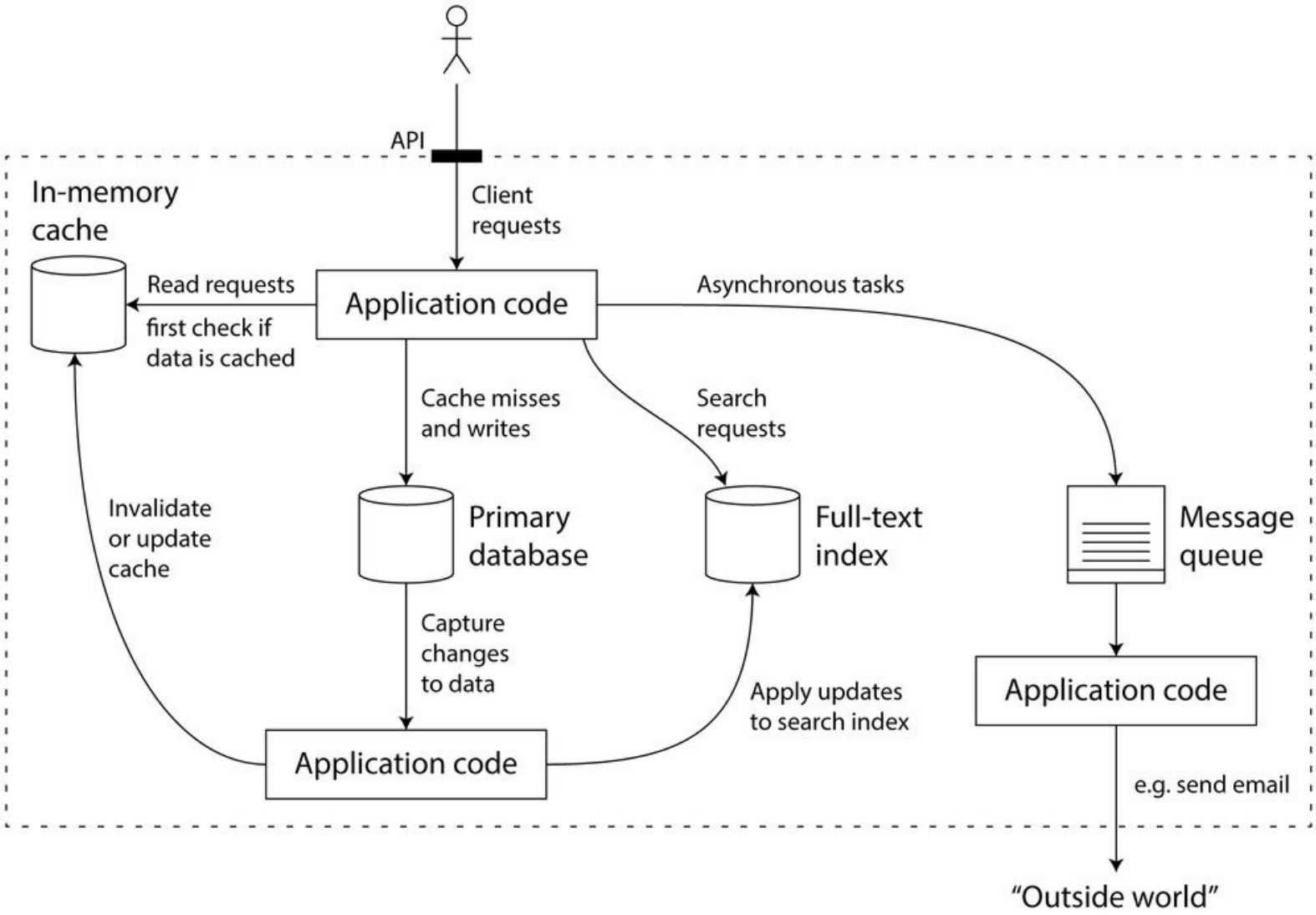
- Store data so that they, or another application, can find it again later (**databases**). → armazenam os dados para utilizações futuras
- Remember the result of an expensive operation, to speed up reads (**caches**). → guardam o resultado de operações dispendiosas
- Allow users to search data by keyword or filter it in various ways (**search indexes**). → permitem filtrar dados
- Send a message to another process, to be handled asynchronously (**message queues**). → comunicação assíncrona entre processos
- Observe what is happening, and act on events as they occur (**stream processing**). → processamento em tempo real, assim que geados
- Periodically crunch a large amount of accumulated data (**batch processing**).

↳ processamentos de dados gerados num determinado Δt.

Thinking about Data Systems

- Increasingly, many applications have wide-ranging requirements # cada vez mais aplicações precisam de mais features
  - 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).

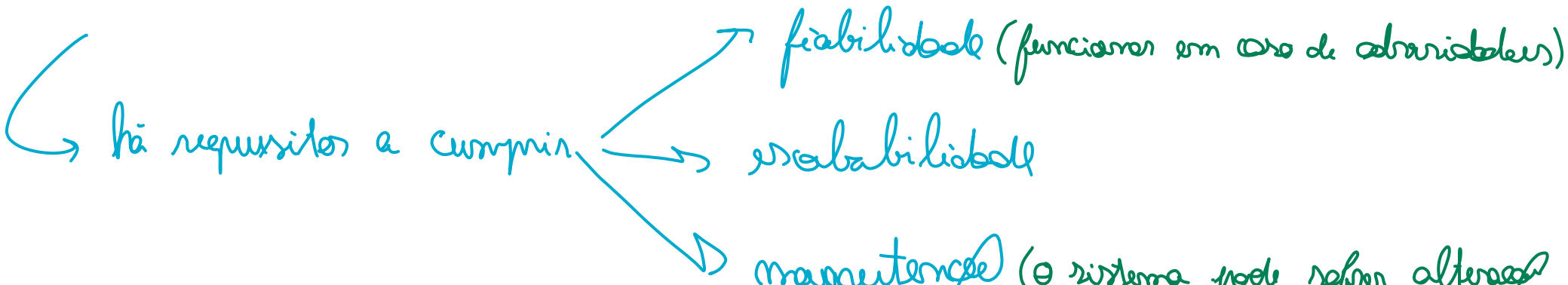
Thinking about Data Systems



// broken into different tasks

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

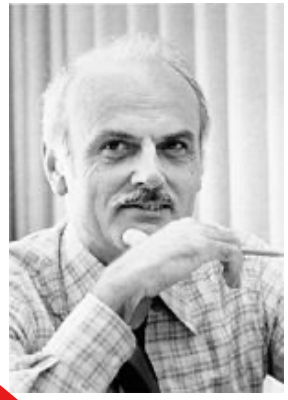
- **Reliability**: The system should continue performing the correct function at the desired performance,
  - even in the face of adversity (hardware or software faults, and even human error).
- **Scalability**: As the system grows (in data volume, traffic volume or complexity), there should be reasonable 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 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 OOPL
  - Persistent types in C++, Java or Small Talk
  - Issues: Lack of high level QL, no standards, performance



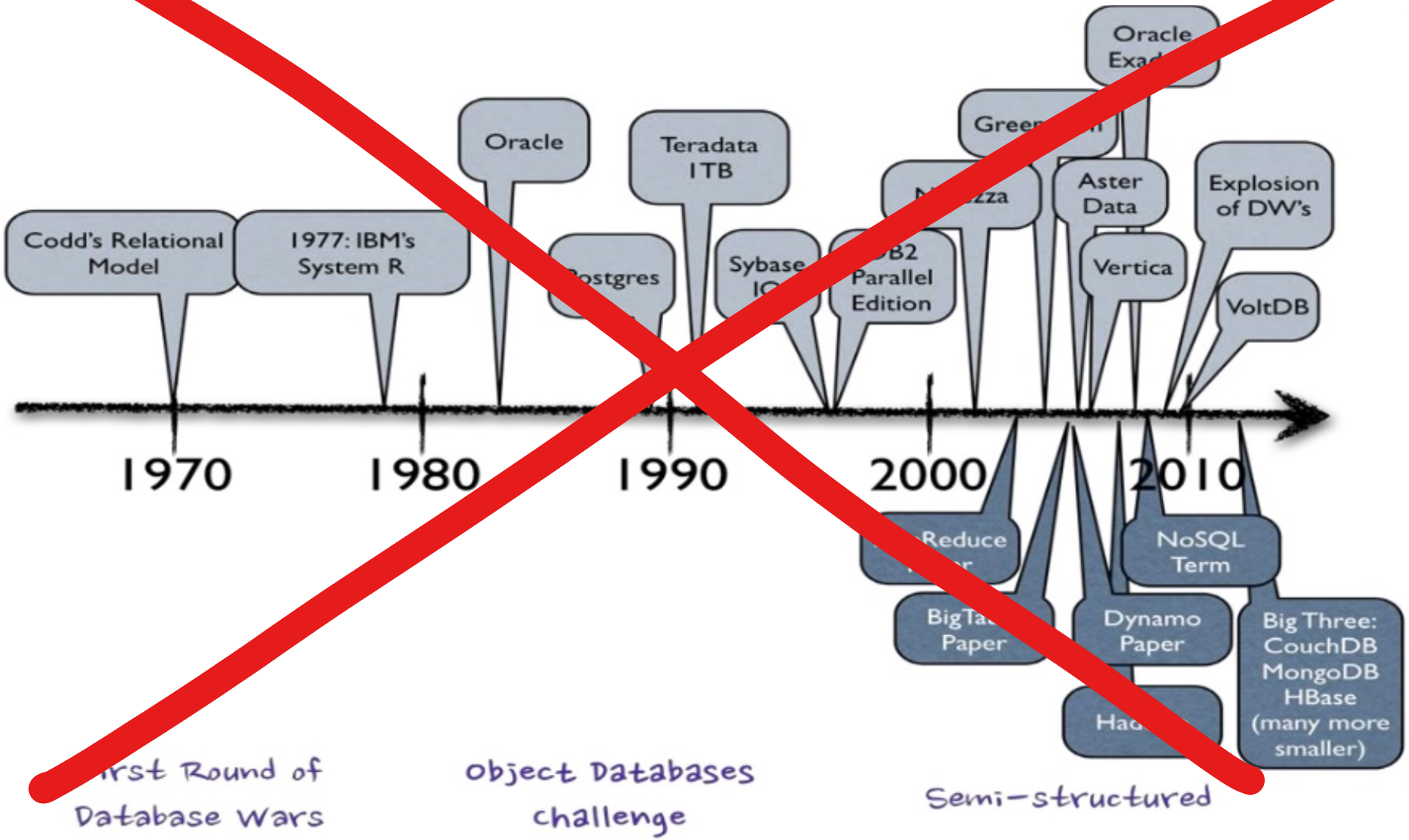
Brief History of Database Systems

- Object-relational DBMS (OR-DBMS) (1990s)
  - Relational DBMS vendors' answer to OO
  - User-defined types, functions (spatial, multimedia) Nested tables
  - SQL: 1999 (2003) standard. 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

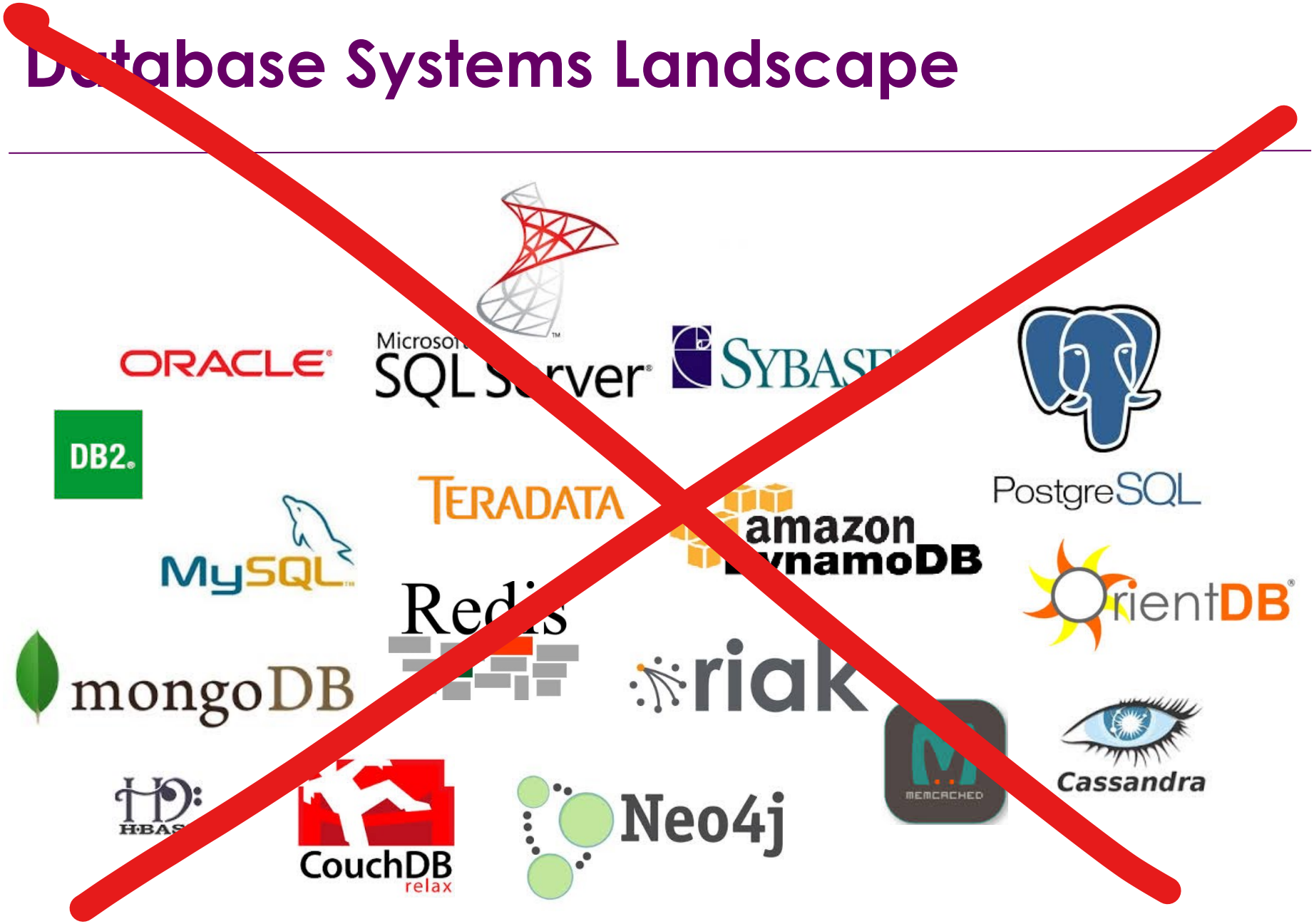
- Data stream management systems (2000s)
  - Continuous query against data streams
- The era of big data (mid 2000-now):
  - **Big data**: datasets that grow so large (terabytes to petabytes) that they become awkward to work with traditional DBMS
  - Parallel DBMSs continue to push the scale of data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

Database Evolution Timeline

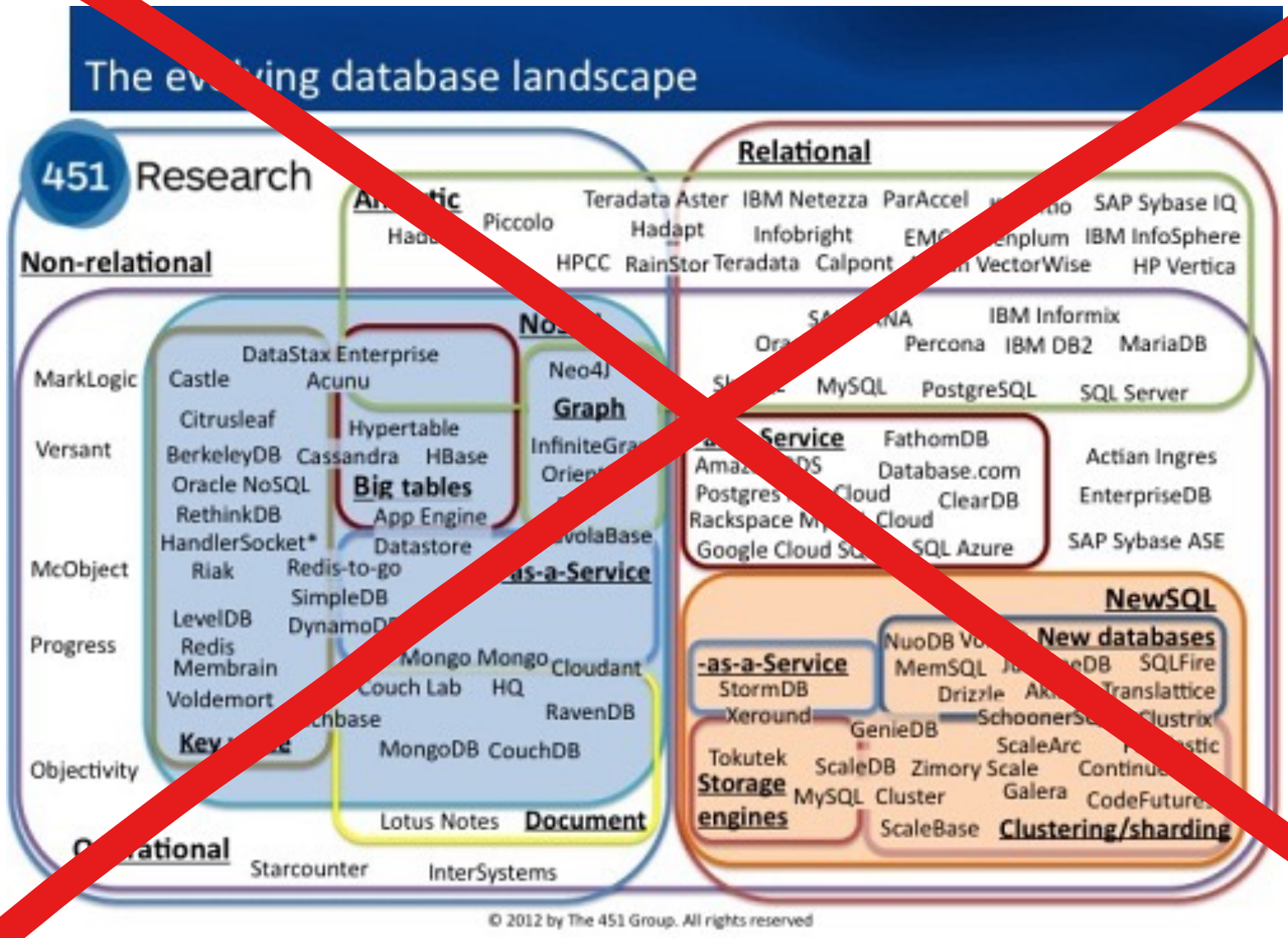


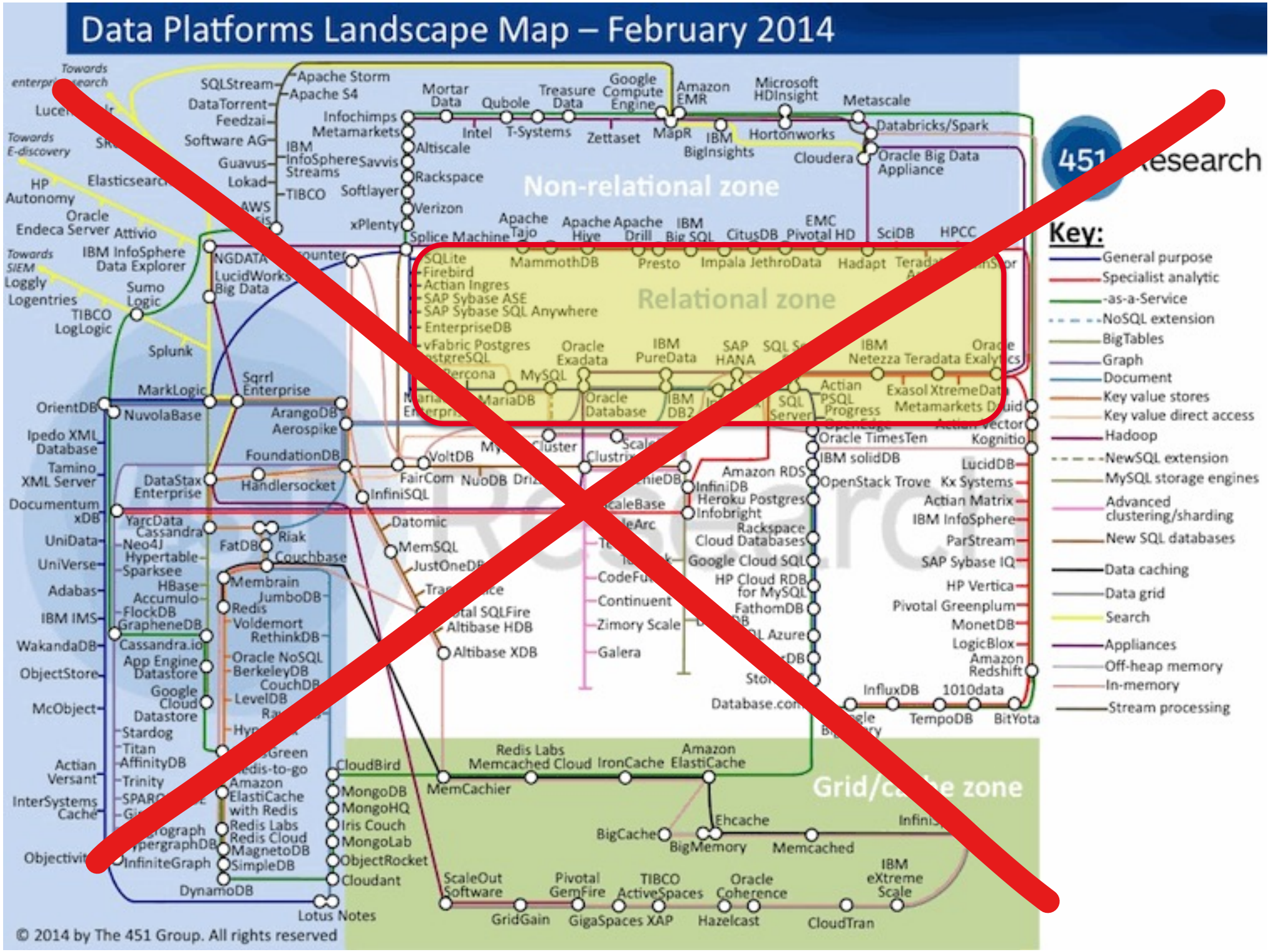
"Big data – an introduction", Peter Morgan

Database Systems Landscape



Database Systems Landscape




Data Platforms Landscape Map – February 2014
Non-relational zone
Relational zone
Grid/cache zone
451 Research
Key:
General purpose
Specialist analytic
-as-a-Service
NoSQL extension
BigTables
Graph
Document
Key value stores
Key value direct access
Hadoop
NewSQL extension
MySQL storage engines
Advanced clustering/sharding
New SQL databases
Data caching
Data grid
Search
Appliances
Off-heap memory
In-memory
Stream processing
Towards enterprise search
Towards E-discovery
Towards SIEM
SQLStream
Apache Storm
Apache S4
DataTorrent
Feedzai
Software AG
Guavus
Lokad
IBM InfoSphere Streams
TIBCO
AWS
Mortar Data
Qubole
Treasure Data
Google Compute Engine
Amazon EMR
Microsoft HDInsight
Metascale
Databricks/Spark
Infochimps
Metamarkets
Altiscale
Rackspace
Softlayer
Verizon
xPlenty
Intel
T-Systems
Zettaset
MapR
IBM BigInsights
Hortonworks
Cloudera
Oracle Big Data Appliance
Splice Machine
Apache Tajo
Apache Hive
Apache Drill
IBM Big SQL
CitusDB
EMC Pivotal HD
SciDB
HPCC
Savvis
Elasticsearch
HP Autonomy
Oracle Endeca Server
Attivio
IBM InfoSphere Data Explorer
Loggly
Logentries
Sumo Logic
TIBCO LogLogic
Splunk
NGDATA
LucidWorks Big Data
SQLite
Firebird
Actian Ingres
SAP Sybase ASE
SAP Sybase SQL Anywhere
EnterpriseDB
vFabric Postgres
PostgreSQL
Percona
MariaDB
MammothDB
Presto
Impala
JethroData
Hadapt
Oracle Exadata
IBM PureData
SAP HANA
MySQL
Oracle Database
IBM DB2
SQL Server
Actian PSQL
Progress
IBM Netezza
Teradata
Exasol
XtremeData
Metamarkets
Kognitio
MarkLogic
Sqrrl Enterprise
OrientDB
NuvolaBase
ArangoDB
Aerospike
FoundationDB
Ipedo XML Database
Tamino XML Server
DataStax Enterprise
Handlersocket
Documentum xDB
VoltDB
FairCom
NuoDB
InfiniSQL
Oracle TimesTen
IBM solidDB
Amazon RDS
InfiniDB
Heroku Postgres
Infobright
Rackspace Cloud Databases
Google Cloud SQL
HP Cloud RDB for MySQL
FathomDB
Azure
Database.com
OpenStack Trove
LucidDB
Kx Systems
Actian Matrix
IBM InfoSphere
ParStream
SAP Sybase IQ
HP Vertica
Pivotal Greenplum
MonetDB
LogicBlox
Amazon Redshift
InfluxDB
1010data
TempoDB
BitYota
YarcData
Cassandra
Neo4J
Hypertable
Sparksee
HBase
Accumulo
FlockDB
GrapheneDB
Cassandra.io
App Engine Datastore
Google Cloud Datastore
Stardog
Titan
AffinityDB
Trinity
SPARQ
Riak
FatDB
Couchbase
Membrain
JumboDB
Redis
Voldemort
RethinkDB
Oracle NoSQL
BerkeleyDB
CouchDB
LevelDB
Datomic
MemSQL
JustOneDB
Altibase HDB
Altibase XDB
CodeFutures
Continuent
Zimory Scale
Galera
UniData
UniVerse
Adabas
IBM IMS
WakandaDB
ObjectStore
McObject
Actian Versant
InterSystems Cache
Objectivity
InfiniteGraph
DynamoDB
Lotus Notes
Redis-to-go
Amazon ElastiCache with Redis
Redis Labs
Redis Cloud
MagnetoDB
SimpleDB
CloudBird
MongoDB
MongoHQ
Iris Couch
MongoLab
ObjectRocket
Cloudant
Redis Labs Memcached Cloud
IronCache
Amazon ElastiCache
MemCachier
BigCache
Ehcache
BigMemory
Memcached
ScaleOut Software
Pivotal GemFire
TIBCO ActiveSpaces
Oracle Coherence
IBM eXtreme Scale
GridGain
GigaSpaces XAP
Hazelcast
CloudTran
© 2014 by The 451 Group. All rights reserved

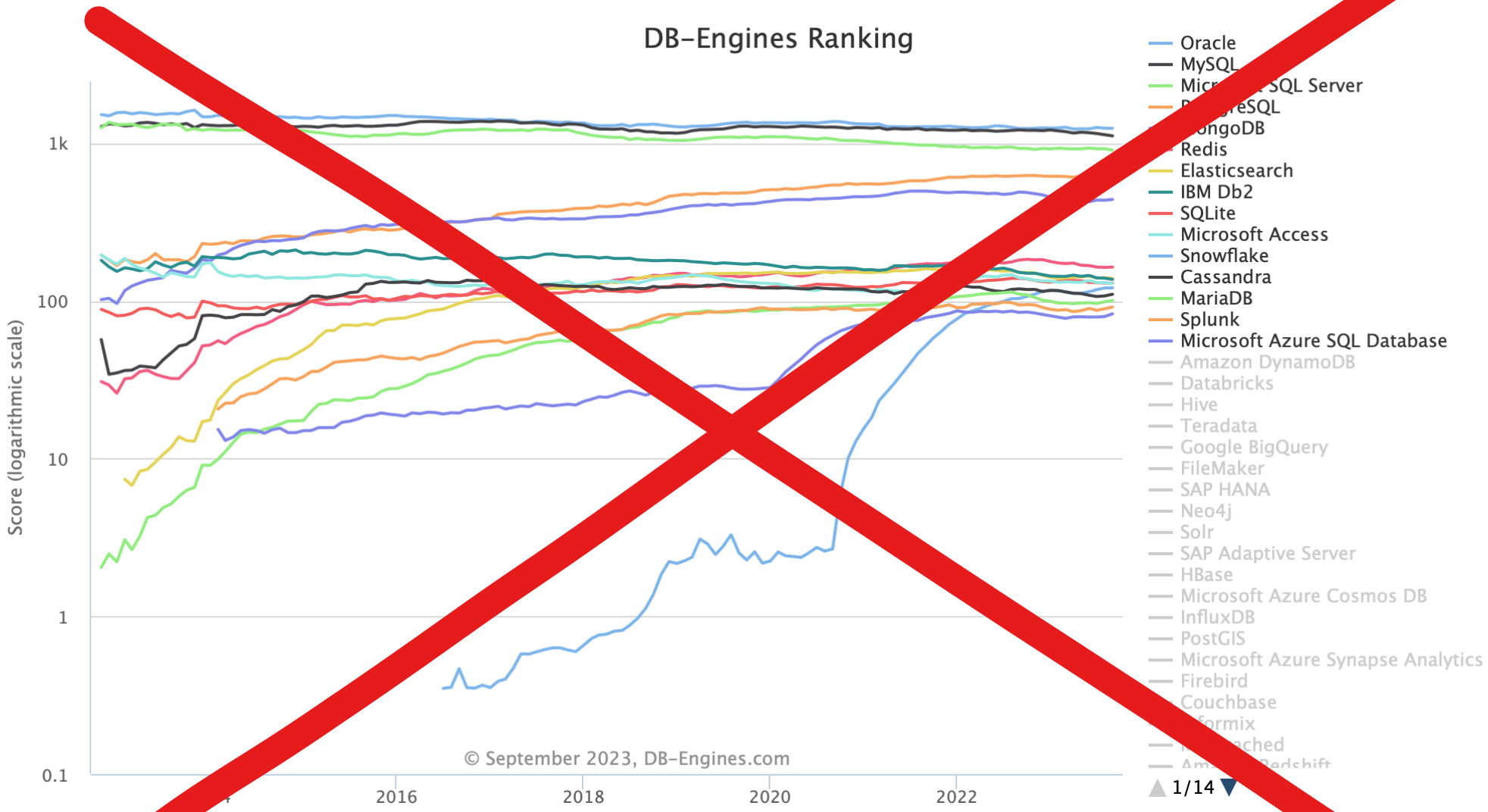
Database Systems Landscape

422 systems in ranking, Sep[illegible]ber 2023

Rank Sep 2023	Rank Aug 2023	Rank Sep 2022	DBMS	Database Model	Score Sep 2023	Score Aug 2023	Score Sep 2022
1.	1.	1.	Oracle	Relational, Multi-model	1240.88	-1.22	+2.62
2.	2.	2.	MySQL	Relational, Multi-m[illegible]	1111.49	-18.97	-100.98
3.	3.	3.	Microsoft SQL Se[illegible]	Relational, M[illegible]model	902.22	-18.60	-24.08
4.	4.	4.	PostgreSQL	Relati[illegible] Multi-model	620.75	+0.37	+0.29
5.	5.	5.	MongoDB	[illegible]ment, Multi-model	439.42	+4.93	-50.21
6.	6.	6.	Redis	Key-value, Multi-model	163.68	+0.72	-17.79
7.	7.	7.	Elasticsearch	Search engine, Multi-model	138.98	-0.94	-12.46
8.	8.	8.	IBM Db2	Relational, Multi-model	136.72	-2.52	-14.67
9.	↑ 10.	↑ 10.	SQLite	[illegible]tional	129.20	-0.72	-9.62
10.	↓ 9.	↓ 9.	Microsoft Access	Rela[illegible]	128.56	-1.78	-11.47
11.	11.	↑ 13.	Snowflake	Relational	120.89	+0.27	+17.39
12.	12.	↓ 11.	Cassandra	Wide column, [illegible]model	110.06	+2.67	-9.06
13.	13.	↓ 12.	MariaDB	Relational, Multi-mo[illegible]	100.45	+1.80	-9.70
14.	14.	14.	Sp[illegible]	Search engine	91.40	+2.42	-2.65
15.	↑ 16.	↑ 16.	[illegible]crosoft Azure SQL Database	Relational, Multi-model	82.73	+3.22	-1.69
16.	↓ 15.	[illegible]	Amazon DynamoDB	Multi-model	[illegible]91	-2.64	-6.51
17.	↑ 18	↑ 20.	Databricks	Multi-model	75.1[illegible]	[illegible]3.84	+19.56
18.	[illegible]7.	↓ 17.	Hive	Relational	71.83	[illegible]	-6.60
[illegible]	19.	↓ 18.	Teradata	Relational, Multi-model	60.33	-0.98	[illegible]6.25
20.	20.	↑ 24.	Google BigQuery	Relational	56.46	+2.56	[illegible]

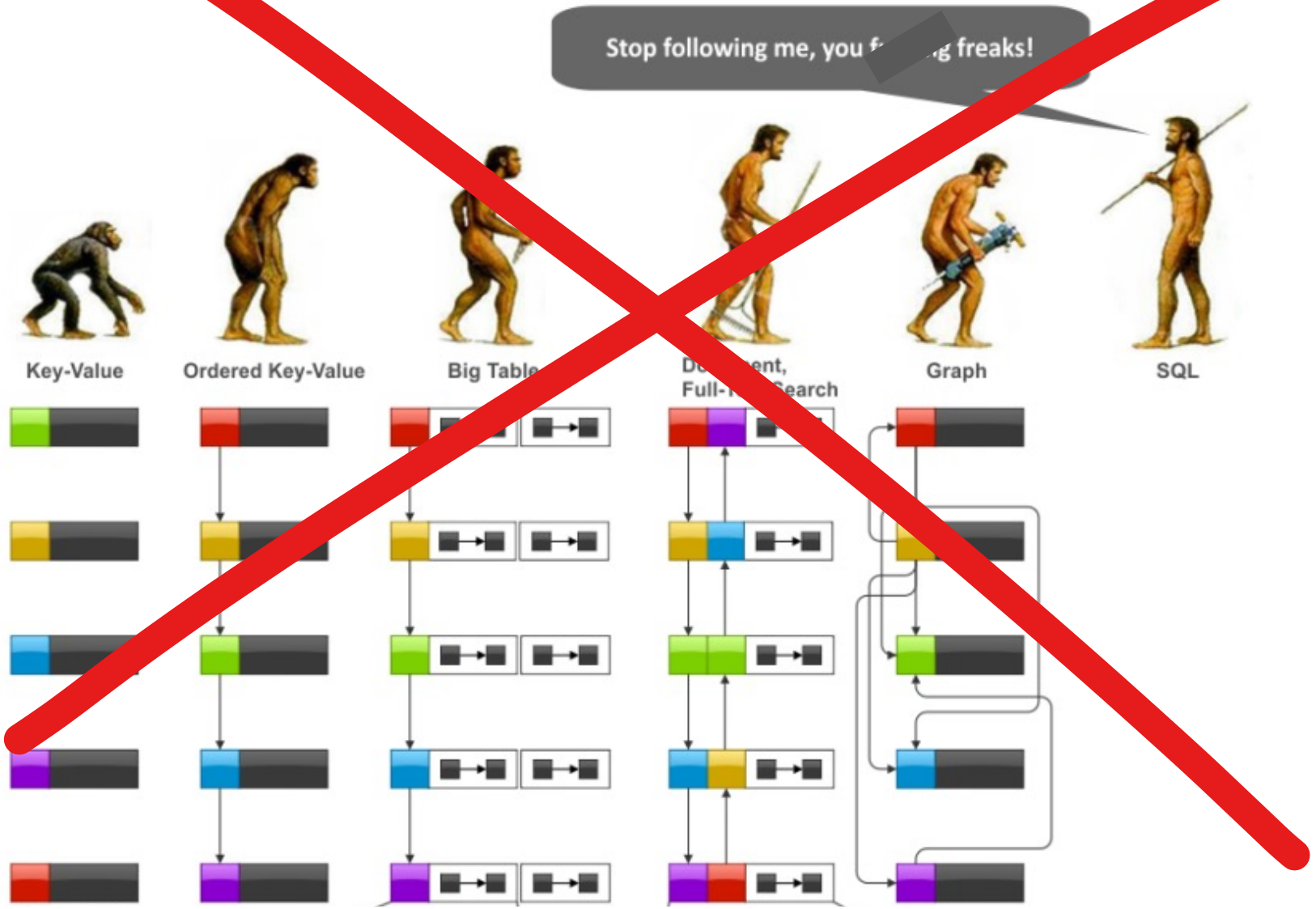
https://db-engines.com/en/ranking

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.