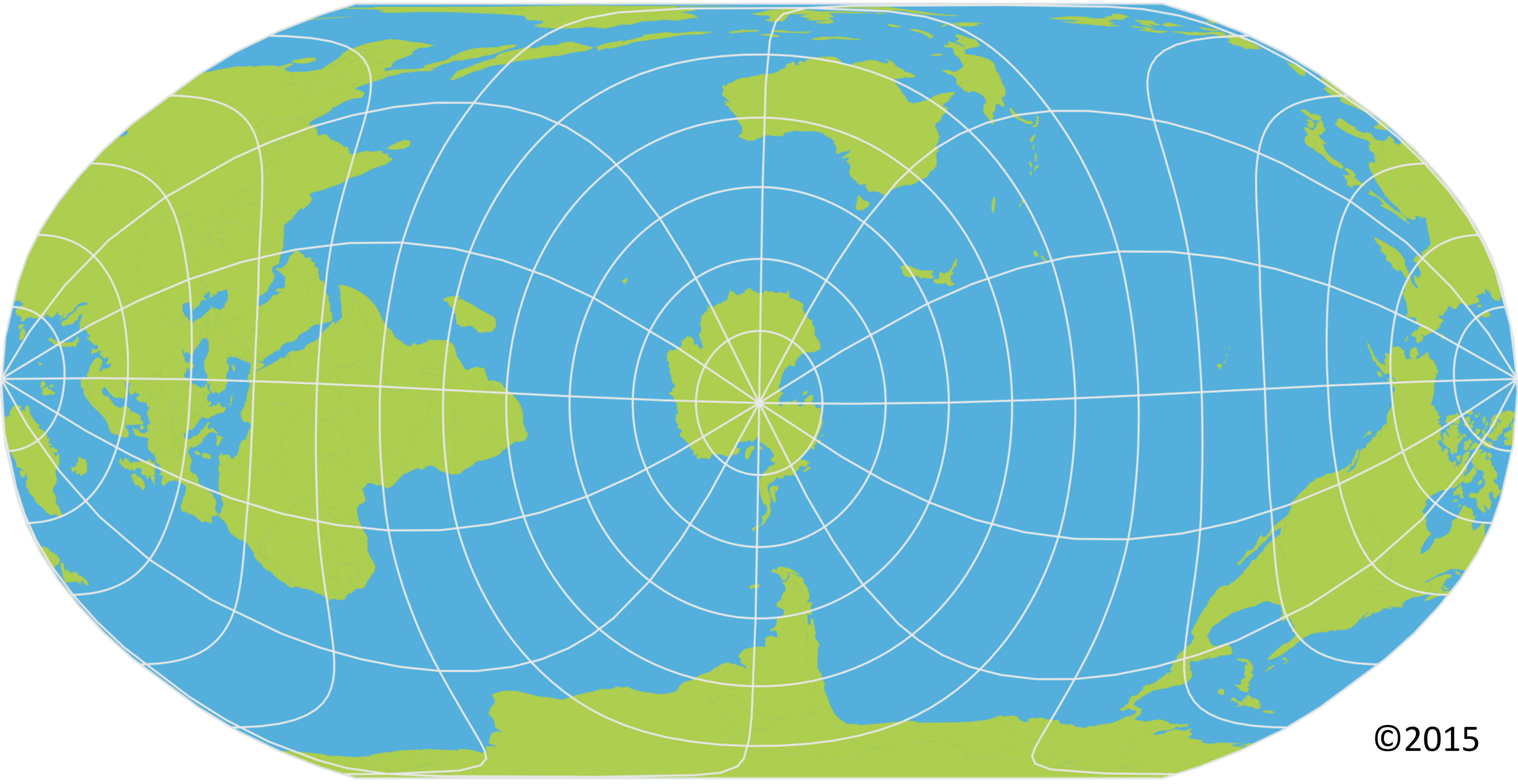
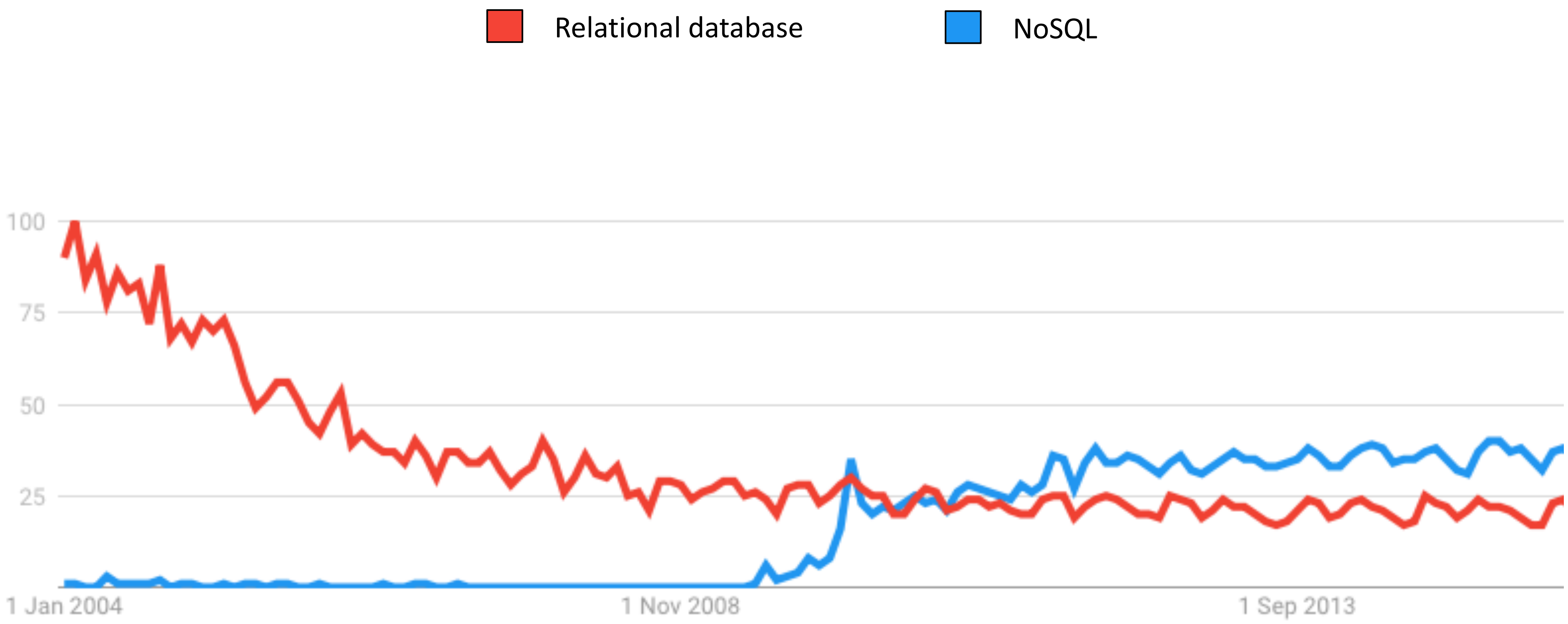
COS 787 – Guidelines for Selecting a Database

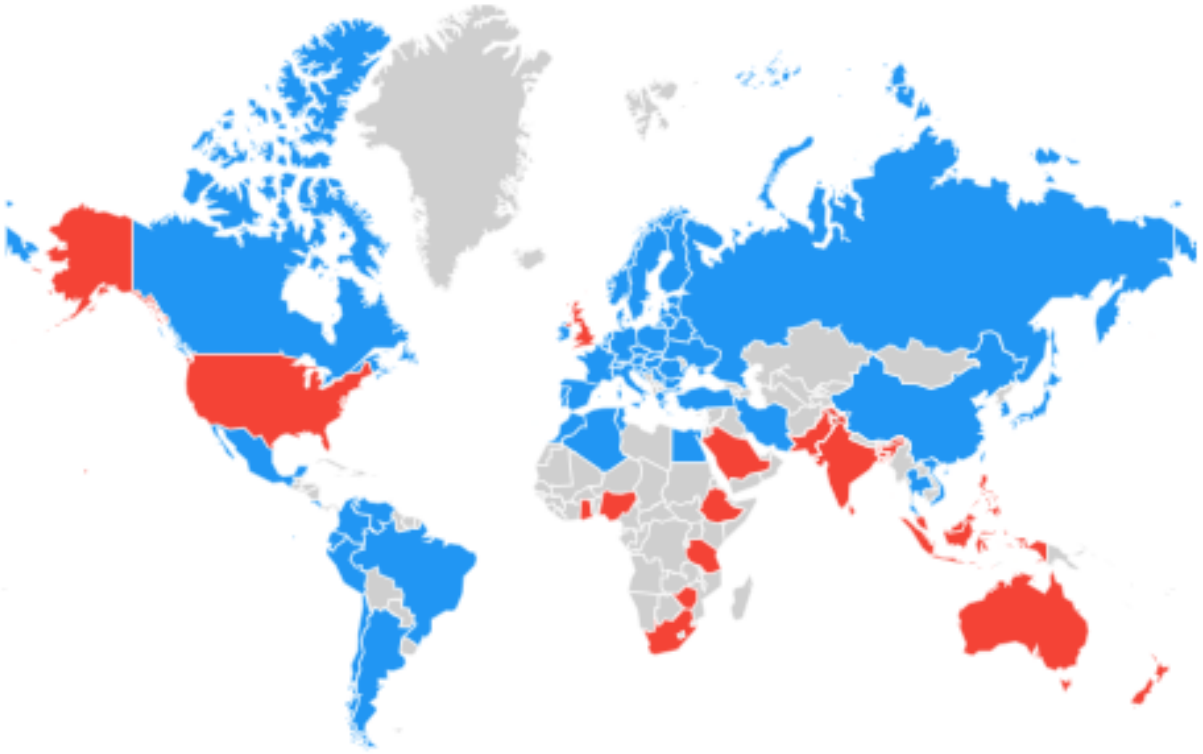
V Rautenbach



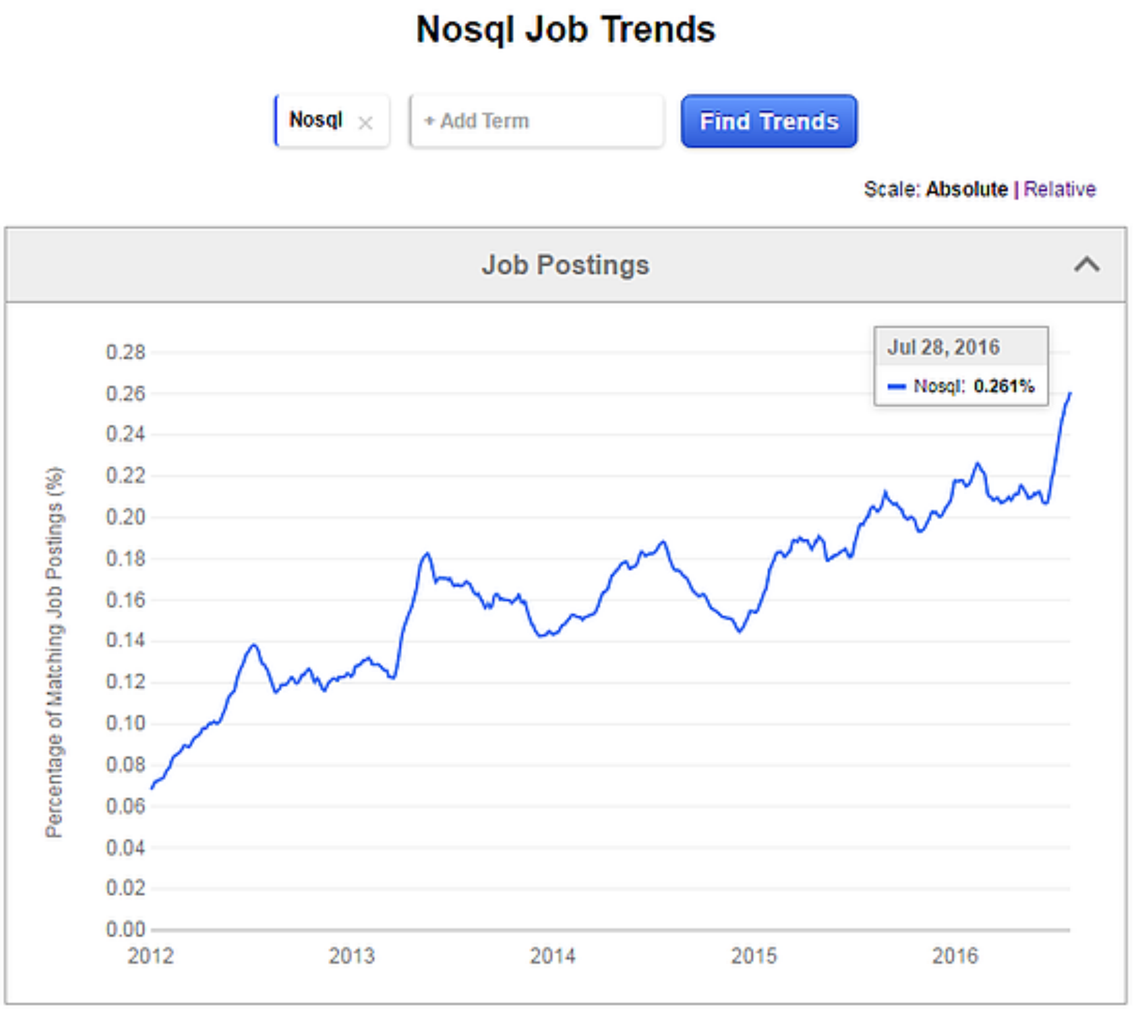
NoSQL
YOU HAD A FAST, BEAUTIFUL HORSE
BUT YOU HAVE NO IDEA WHERE IT IS



https://trends.google.com/trends/explore?date=all&q=nosql,relational%20database



https://trends.google.com/trends/explore?date=all&q=nosql,relational%20database


Nosql Job Trends
Nosql
+ Add Term
Find Trends
Scale: Absolute | Relative
Job Postings
Jul 28, 2016
Nosql: 0.261%
Percentage of Matching Job Postings (%)
0.28
0.26
0.24
0.22
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
2012
2013
2014
2015
2016

Interview Questions

1. What is the difference between NoSQL and RDBMS? Why and why not to use NoSQL databases? What are the various advantages of NoSQL databases?
2. When should I use a NoSQL database instead of a relational database?
3. Could you explain the transaction support by using BASE in NoSQL systems?
4. Does NoSQL Database Interact With Oracle Database?
5. Tell me the challenges of using NoSQL.

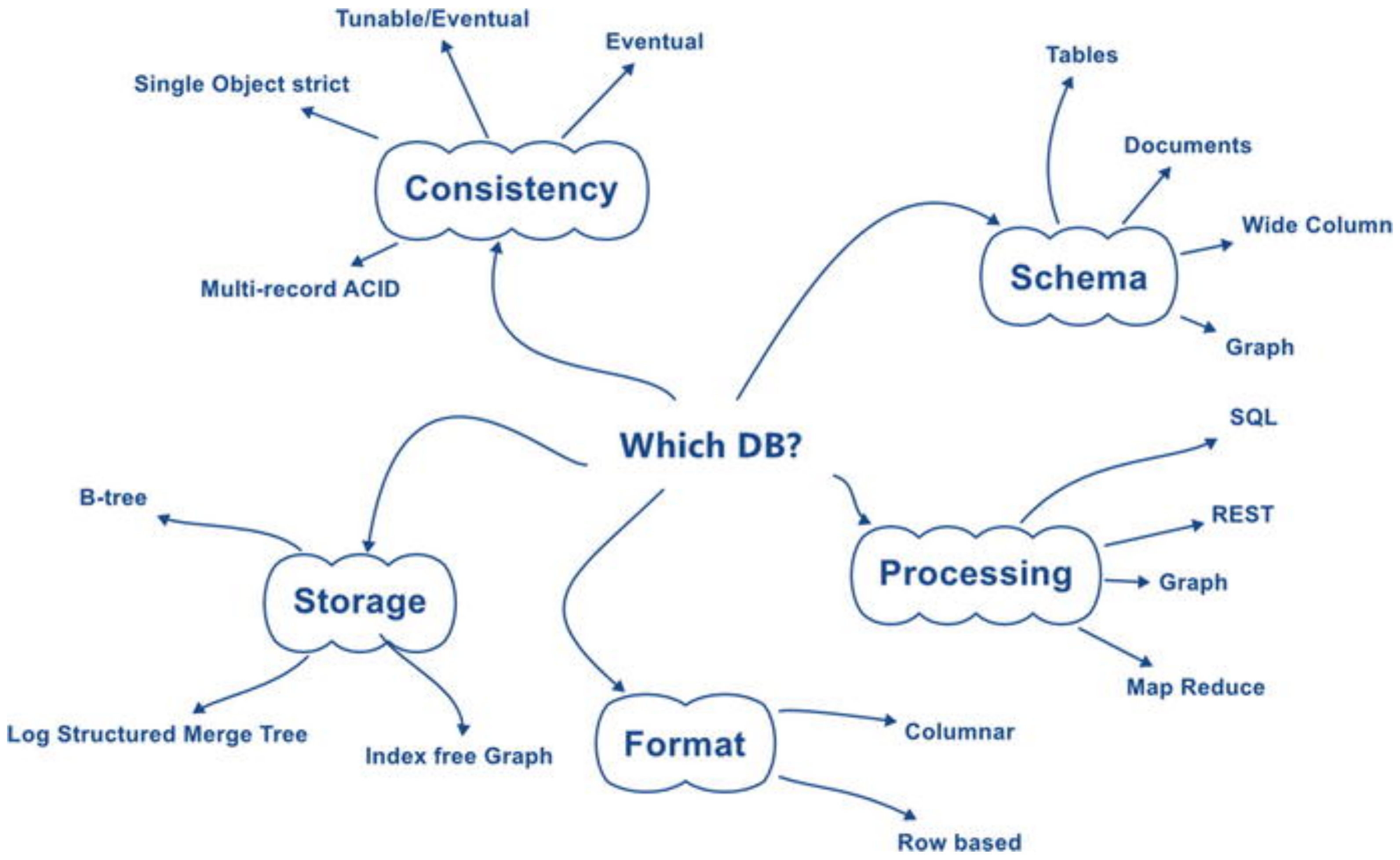
“One’s philosophy is not best expressed in words; it is expressed in the choices one makes.”
- Eleanor Roosevelt

Introduction

- Developers have never had as many good database options as they have today.
  - Relational databases have a **long and proven track** record of **successful use** in a wide range of applications.
  - These databases have been so successful they **virtually eliminated** the widespread use of **earlier database models**, such as file-based, hierarchical, and network databases.
  - It was not until the advent of **commercial web systems**, such as search engines, that **relational databases strained to meet developers' demands**.

Introduction

- In relational database design, the **structure and relations of entities** drives design—not so in NoSQL database design.
  - Of course, you will model entities and relations, but performance is more important than preserving the relational model.
- The **relational model** emerged for **pragmatic reasons**, that is, data anomalies and **difficulty reusing** existing databases for new applications.
- **NoSQL** databases also emerged for **pragmatic reasons**, specifically, the **inability to scale** to meet growing demands for high volumes of read and write operations.
- In exchange for improved read and write performance, you may **lose other features of relational databases**, such as immediate consistency and ACID transactions (although this is not always the case).


Which DB?
Consistency
Tunable/Eventual
Eventual
Single Object strict
Multi-record ACID
Schema
Tables
Documents
Wide Column
Graph
Processing
SQL
REST
Graph
Map Reduce
Storage
B-tree
Log Structured Merge Tree
Index free Graph
Format
Columnar
Row based

Criteria for Selecting Relational databases

- Relational databases is known for their **reliability and maintaining data integrity**.
- Reasons for selecting RDBMs over the newer types:
  - Relational database **normalization** means that each fact in your data is stored only once, so you shouldn't get data anomalies.
  - Relational schema means you always know what **columns (i.e. attributes) exist for a given row**.
    - Some people find it limiting to obey a schema, but 99% of the time, it's easier than writing lots of application code to do the same checks.
  - **Data types** help to ensure data in a given column is well-formed.
    - CHECK constraints can further validate data

Criteria for Selecting Relational databases

- Reasons for selecting RDBMs over the newer types:
  - **Enforcement of declarative constraints**, such as NOT NULL, UNIQUE KEY, or FOREIGN KEY
    - yes, you can benefit from foreign keys even though you don't use joins in your queries
  - **Authentication and access privileges** are handled better in SQL
    - Security is still pretty primitive in many NoSQL products (this varies; some are more capable than others)

Criteria for Selecting Relational databases

- Reasons for selecting RDBMs over the newer types:
  - **SQL** is the dominant query language for relational databases
    - SQL is an industry standard, interoperable between platforms and application programming languages, well-documented, and stable
  - Many RDBMS exist that have solved the **ACID** requirement with a good balance between **durability and performance**
    - The performance of NoSQL databases tends to plummet if you configure them to have durability matching that of an RDBMS

Criteria for Selecting Key-Value Databases

- Key-value databases are well suited to applications that have **frequent small reads and writes** along with simple data models.
- The values stored in key-value databases may be **simple scalar values**, such as integers or Booleans, but they may be **structured data types**, such as lists and JSON structures.
- Key-value databases generally have **simple query facilities** that allow you to look up a value by its key.
- Some key-value databases support **search features** that provide for somewhat more flexibility.

Criteria for Selecting Key-Value Databases

- Key-value databases are used in a wide range of applications, such as the following:
  - **Caching data** from relational databases to improve performance
  - **Tracking transient attributes** in a web application, such as a shopping cart
  - **Storing configuration** and **user data information** for mobile applications
  - **Storing large objects**, such as images and audio files

Criteria for Selecting Document Databases

- Document databases are designed for **flexibility**.
  - If an application requires the **ability to store varying attributes** along with **large amounts of data**, then document databases are a good option.
  - For example, to represent products in a relational database, a modeler may use a table for common attributes and additional tables for each subtype of product to store attributes used only in the subtype of product.
  - Document databases can handle this situation easily.

Criteria for Selecting Document Databases

- Document databases provide for **embedded documents**, which are useful for **denormalizing**.
  - Instead of storing data in different tables, data that is **frequently queried together is stored together in the same document**.
- Document databases **improve on the query capabilities** of **key-value databases** with **indexing** and the ability to filter documents based on attributes in the document.

Criteria for Selecting Document Databases

- Document databases are probably the **most popular** of the NoSQL databases because of their **flexibility**, **performance**, and **ease of use**.
- These databases are well suited to a number of use cases, including
  - **Back-end support for websites** with high volumes of reads and writes
  - Managing **data types with variable attributes**, such as products
  - Tracking variable types of **metadata**
  - Applications that use **JSON data structures**
  - Applications benefiting from **denormalization** by embedding structures within structures

Criteria for Selecting Graph Databases

- Problem domains that lend themselves to representations as **networks of connected entities** are well suited for graph databases.
- Consider examples mentioned in the discussion of graph databases, such as highways connecting cities, proteins interacting with other proteins, and employees working with other employees.
  - There is some type of connection, link, or direct relationship between two instances of entities.
- These are the types of problem domains that are well suited to graph databases.

Criteria for Selecting Graph Databases

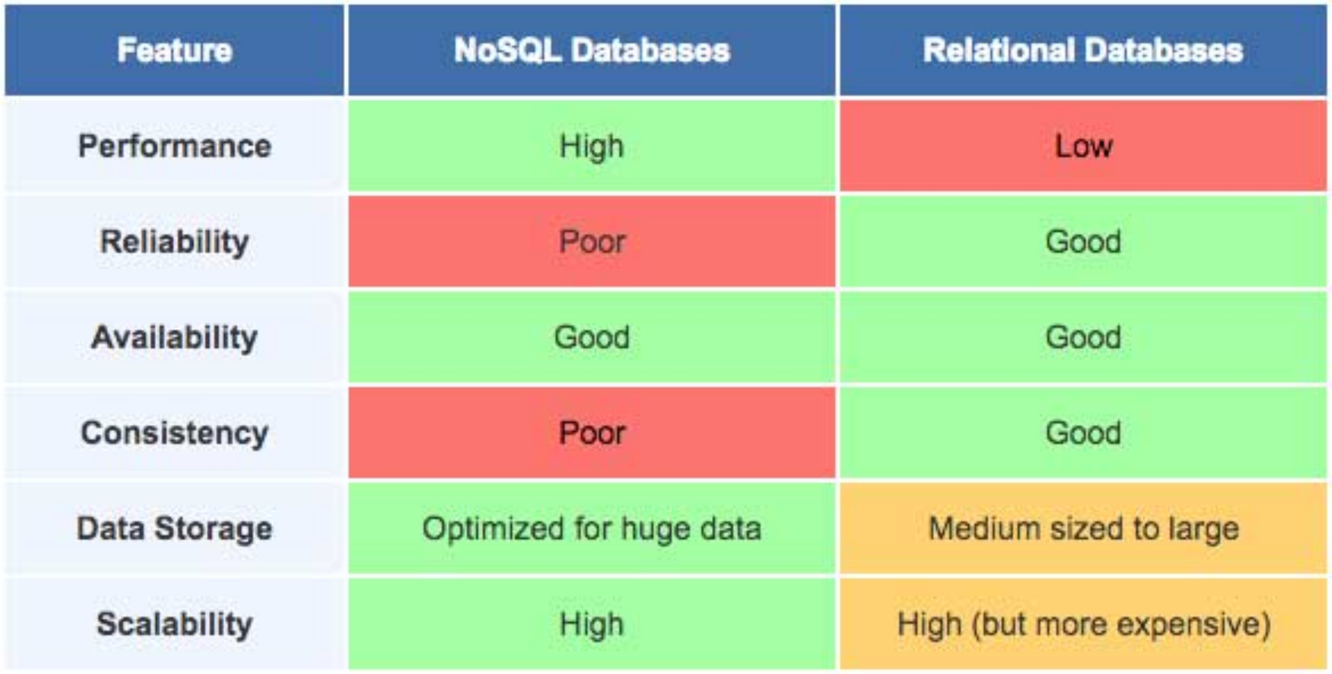
- Other examples of these types of problem domains include
  - Network and IT infrastructure management
  - Identity and access management
  - Business process management
  - Recommending products and services
  - Social networking

Using NoSQL and Relational Databases Together

- **NoSQL and relational databases are complementary.**
- Relational databases offer many features that protect the integrity of data and reduce the risk of data anomalies.
- Relational databases incur operational overhead providing these features.
- In some use cases, performance is more important than ensuring immediate consistency or supporting ACID transactions.
  - In these cases, NoSQL databases may be the better solution. Choosing a database is a process of choosing the right tool for the job.
  - The more varied your set of jobs, the more varied your toolkit.

Final though on selecting a database

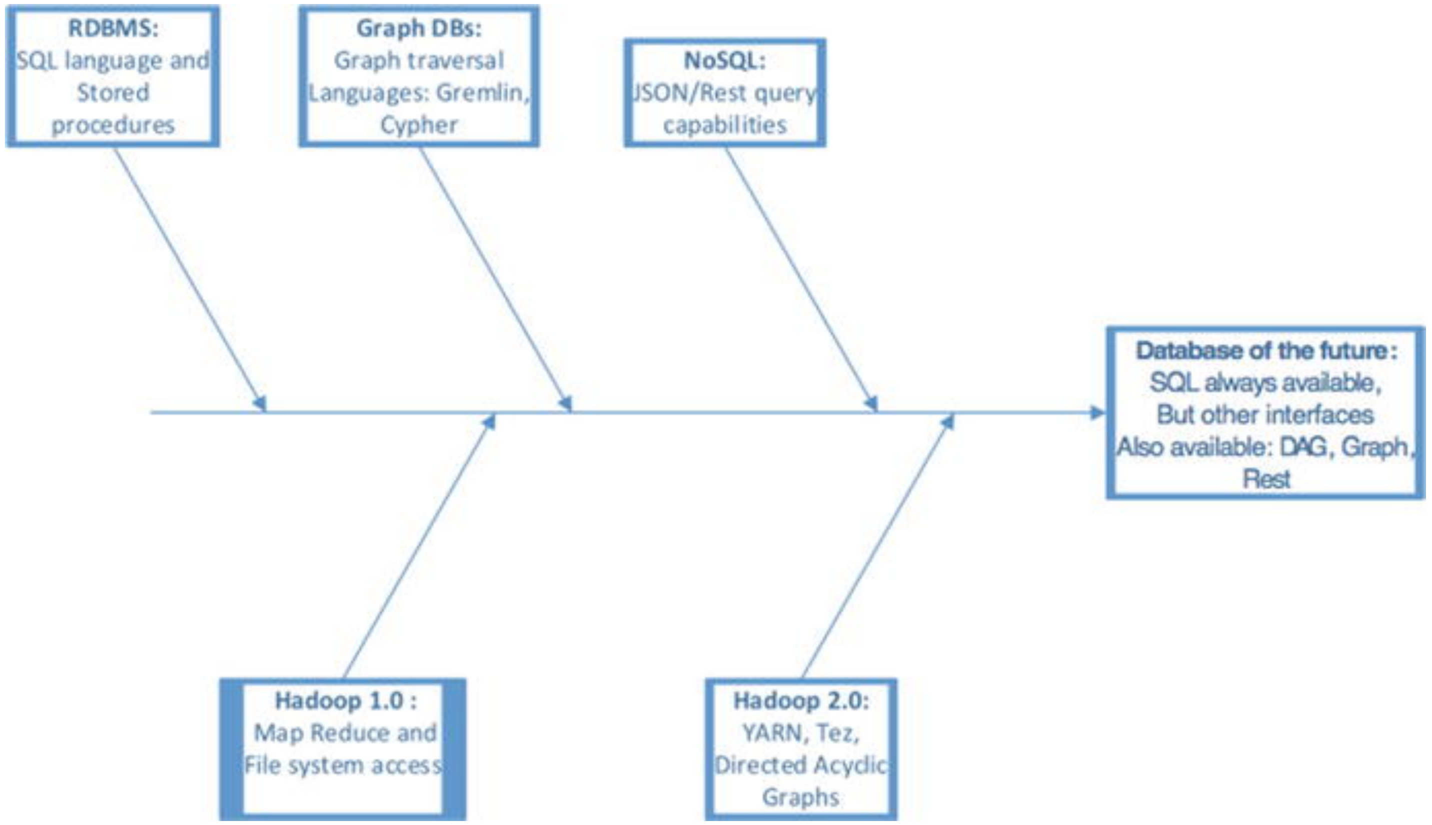
- Just as there is no best programming language, there is no best database management system.
- There are database systems better suited to some problems than others, and the job of developers and designers is to find the best database for the requirements at hand.



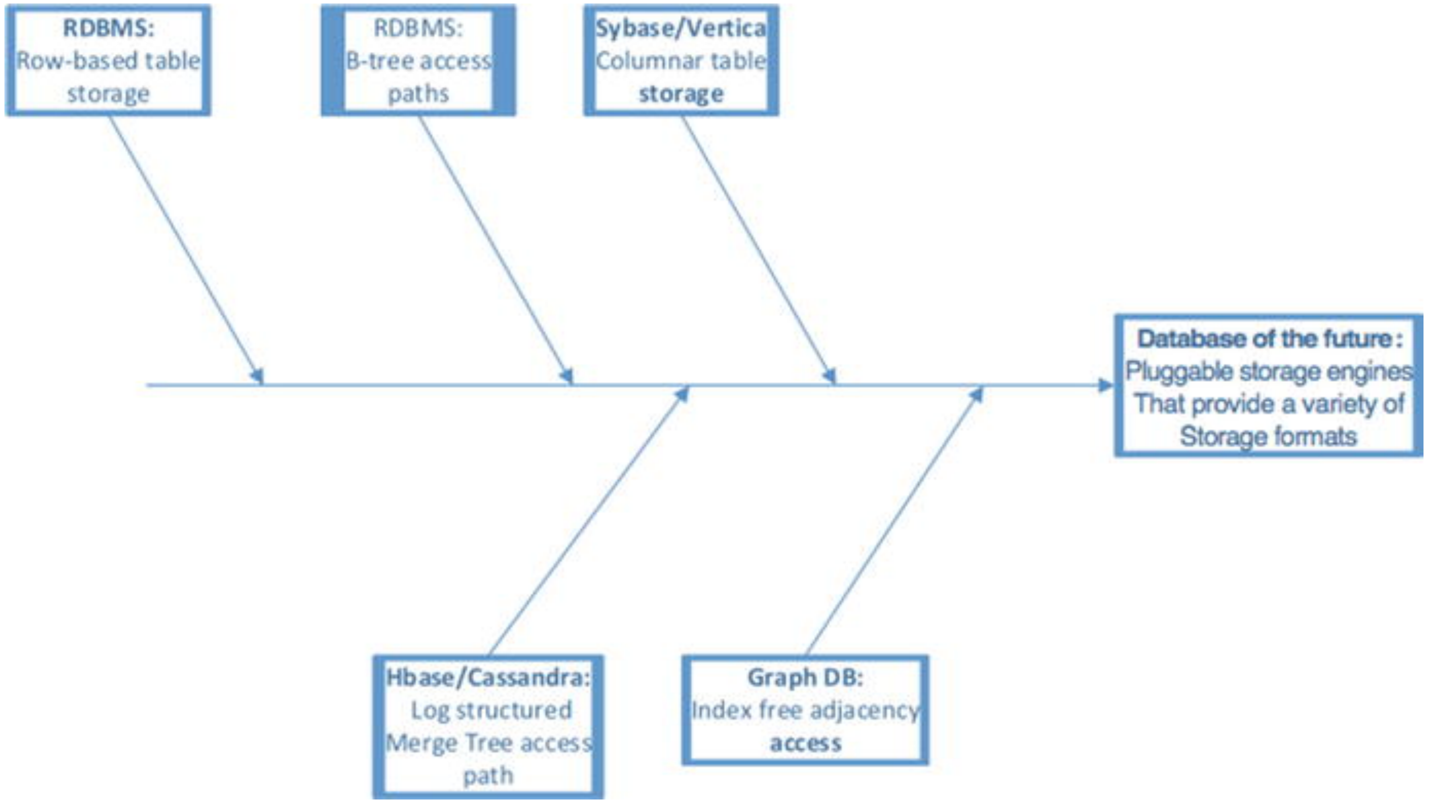
Feature	NoSQL Databases	Relational Databases
Performance	High	Low
Reliability	Poor	Good
Availability	Good	Good
Consistency	Poor	Good
Data Storage	Optimized for huge data	Medium sized to large
Scalability	High	High (but more expensive)

https://blog.pandorafms.org/nosql-databases-the-definitive-guide/

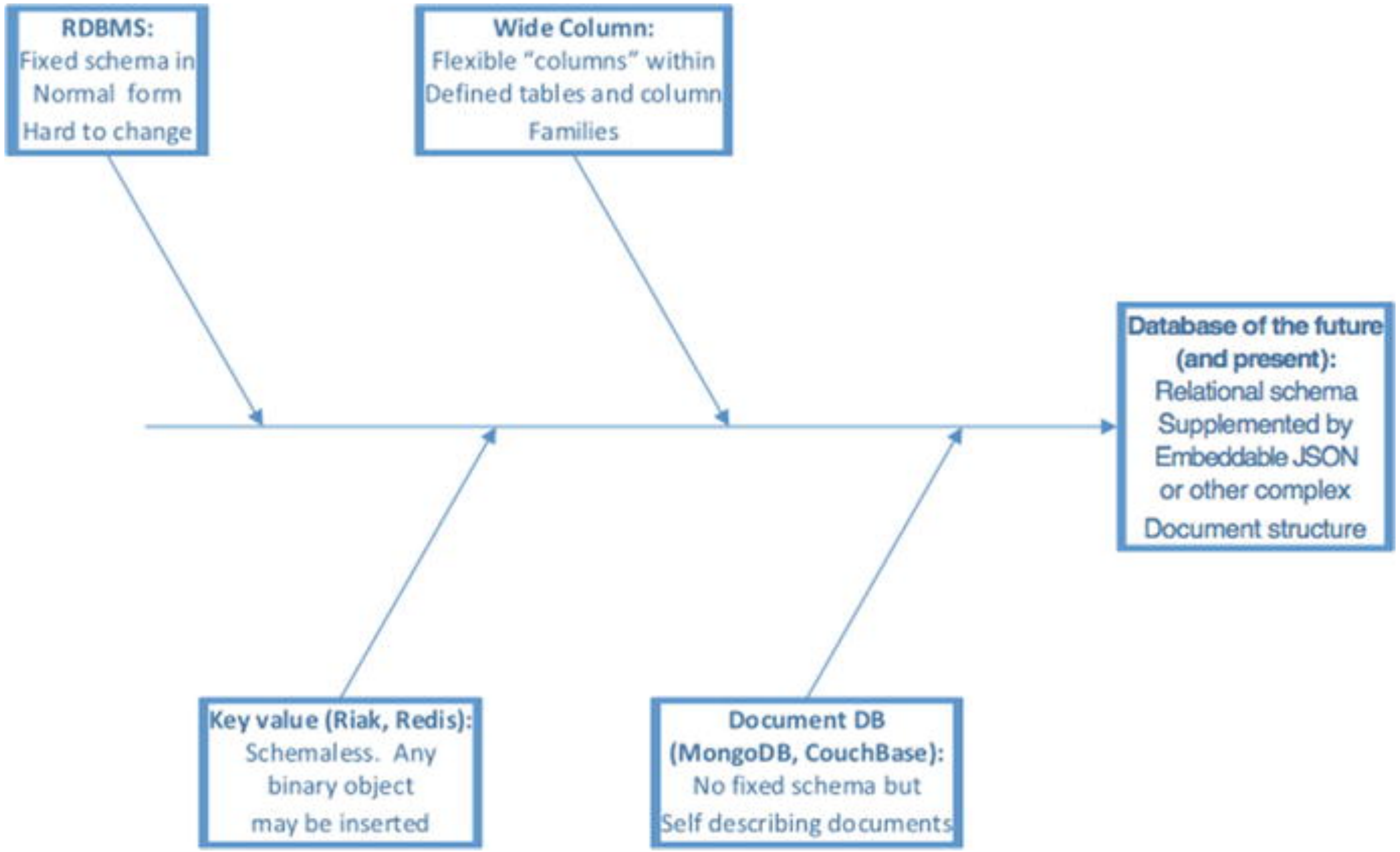
Database languages of the future


RDBMS: SQL language and Stored procedures
Graph DBs: Graph traversal Languages: Gremlin, Cypher
NoSQL: JSON/Rest query capabilities
Database of the future: SQL always available, But other interfaces Also available: DAG, Graph, Rest
Hadoop 1.0 : Map Reduce and File system access
Hadoop 2.0: YARN, Tez, Directed Acyclic Graphs

Storage


RDBMS: Row-based table storage
RDBMS: B-tree access paths
Sybase/Vertica Columnar table storage
Hbase/Cassandra: Log structured Merge Tree access path
Graph DB: Index free adjacency access
Database of the future: Pluggable storage engines That provide a variety of Storage formats

Schemas


RDBMS:
Fixed schema in
Normal form
Hard to change
Wide Column:
Flexible "columns" within
Defined tables and column
Families
Database of the future
(and present):
Relational schema
Supplemented by
Embeddable JSON
or other complex
Document structure
Key value (Riak, Redis):
Schemaless. Any
binary object
may be inserted
Document DB
(MongoDB, CouchBase):
No fixed schema but
Self describing documents

Disruptive Database Technologies

- Storage Technologies
- Blockchains
- Quantum Computing

Relational databases are being phased out as more and more companies make the switch to NoSQL technologies.

- True
- False

Relational databases are being phased out as more and more companies make the switch to NoSQL technologies.

- True
- **False**

Which of the following statements is true of NoSQL databases?

- They don't support any Structured Query Language (SQL) functions.
- They are useful for managing large sets of distributed data.
- They are the most commonly used databases today.
- They cannot be used in conjunction with relational databases.

Which of the following statements is true of NoSQL databases?

- They don't support any Structured Query Language (SQL) functions.
- **They are useful for managing large sets of distributed data.**
- They are the most commonly used databases today.
- They cannot be used in conjunction with relational databases.

In terms of data security, NoSQL databases don't offer the same level of security as RDBMSs due to:

- The lack of integrated features for data confidentiality, integrity and availability
- The lack of ACID compliance for guaranteeing transactional reliability
- The lack of default-enabled authentication and authorization features in many technologies
- All of the above

In terms of data security, NoSQL databases don't offer the same level of security as RDBMSs due to:

- The lack of integrated features for data confidentiality, integrity and availability
- The lack of ACID compliance for guaranteeing transactional reliability
- The lack of default-enabled authentication and authorization features in many technologies
- **All of the above**

While NoSQL databases avoid the rigid schemas of relational databases, the types of NoSQL technologies vary and can be separated into the following primary categories:

- Document databases, graph databases, key-value databases and wide column stores
- CouchDB, MongoDB, Cassandra and HBase
- Those that manage data in the cloud and those that don't
- Oracle NoSQL database, NoSQL for Windows Azure and IBM DB2

While NoSQL databases avoid the rigid schemas of relational databases, the types of NoSQL technologies vary and can be separated into the following primary categories:

- **Document databases, graph databases, key-value databases and wide column stores**
- CouchDB, MongoDB, Cassandra and HBase
- Those that manage data in the cloud and those that don't
- Oracle NoSQL database, NoSQL for Windows Azure and IBM DB2

The adoption level of NoSQL software is lower than that of relational databases, data appliances and columnar database software.

- True
- False

The adoption level of NoSQL software is lower than that of relational databases, data appliances and columnar database software.

- **True**
- False

Even for big data environments, consultants recommend evaluating your project needs and selecting the database technology that best suits those requirements, instead of assuming that NoSQL technologies are always a better fit than relational software.

- True
- False

Even for big data environments, consultants recommend evaluating your project needs and selecting the database technology that best suits those requirements, instead of assuming that NoSQL technologies are always a better fit than relational software.

- **True**
- False

Compare NoSQL & RDBMS

Criteria	**RDBMS**	**NoSQL**
Data format		
Scalability		
Querying		
Storage mechanism		

Compare NoSQL & RDBMS

Criteria	**RDBMS**	**NoSQL**
Data format	Organized and structured	Does not follow any order
Scalability	Average	Very Good
Querying	Using SQL	Limited as no Join Clause
Storage mechanism	Data & relationship stored in different tables	Key-Value Pair, document, column storage, etc.

victoria.rautenbach@up.ac.za