

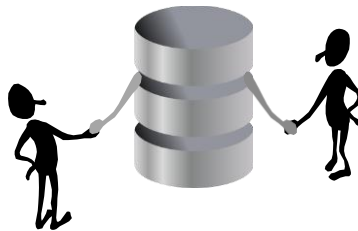
Graph & Modern Databases

COMP1835

1

1

Overview



2

2

Objectives

- ▶ To discuss challenges of RDMS
- ▶ To introduce NoSQL
- ▶ To compare RDMS and NoSQL
- ▶ To discuss BASE vs ACID
- ▶ To introduce CAP theorem

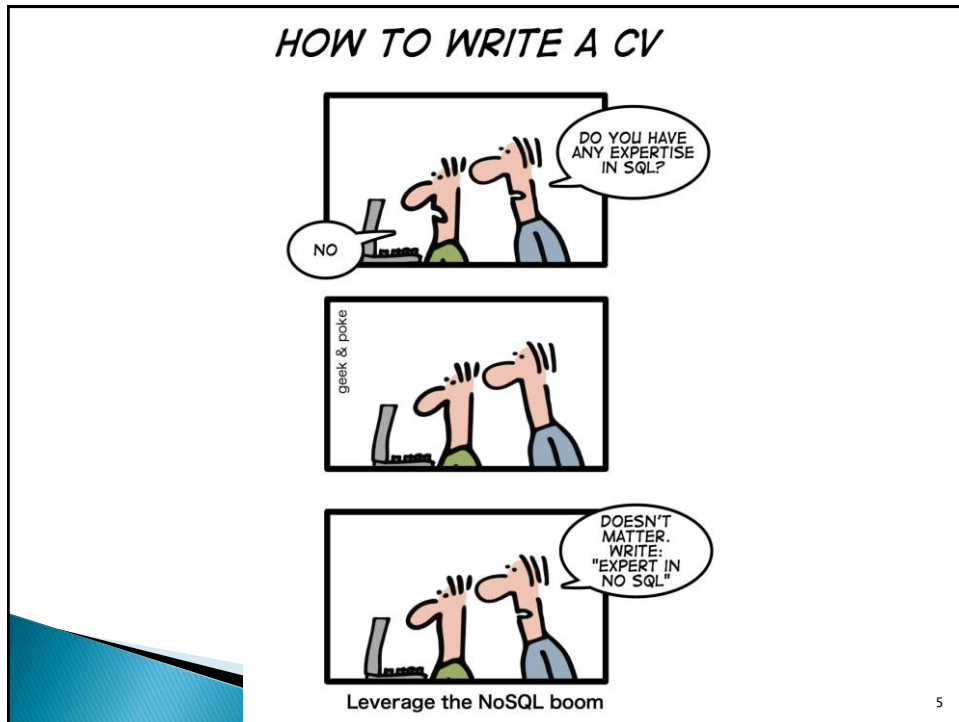
3

3

Part 1

4

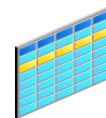
4



5

Structured data

- ▶ For over decades, relational databases have been used to store what we know as structured data:
 - The data is sub-divided into groups, referred to as **tables**.
 - The tables store well-defined units of data in terms of type, size, and other constraints.
 - Each unit of data is known as **column** while each unit of the group is known as **row**.
 - The columns may have relationships defined across tables, for example parent-child, and hence the name relational databases



6

6

Challenges of RDMS

- ▶ RDBMS assumes a well defined structure in data
- ▶ RDBMS assumes that the data is dense and is largely uniform
- ▶ RDBMS builds on a prerequisite that the properties of the data can be defined up front and that its interrelationships are well established
- ▶ With massive data sets the typical storage mechanisms and access methods are getting stretched



7

7

Big Data? How big is Big Data?

- ▶ Google processes more than 20 **petabytes** of data per day!
- ▶ The total size of photos in Facebook is about **100 petabytes** and Facebook generates **4 new petabytes** of data per day!
- ▶ The movie Avatar took up to **1 petabyte** of storage space for the rendering of 3D CGI effects, equivalent to a **32 YEAR long MP3**
- ▶ The EMC report claims that the total size of digital data is doubling in size every two years, and by 2020 the digital universe reached **44 zettabytes**, or **44 trillion gigabytes**.

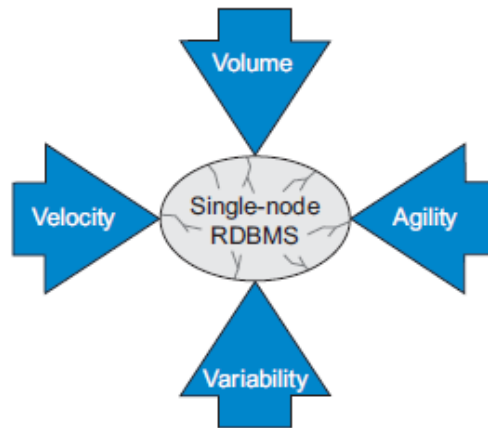
$$1 \text{ Petabyte (PB)} = 10^{15} \text{ b}$$

$$1 \text{ Zeetabyte (ZB)} = 10^{21} \text{ b}$$

8

8

Demands of volume, velocity, variability, and agility

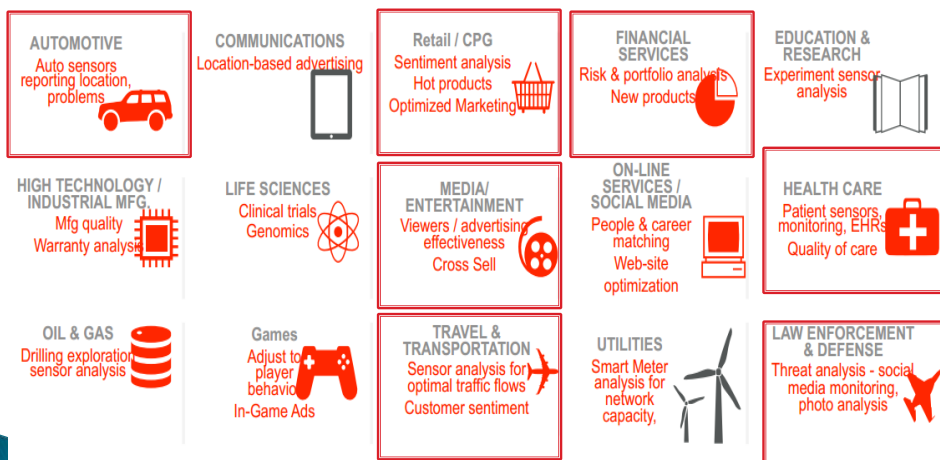


Dan McCreary and Ann Kelly
Making sense of NoSQL 2014.

9

9

Big Data Use Cases



Big Data – The Big Story
Jean-Pierre Dijkstra
Big Data Product Management

10

10

Problems



- ▶ Efficiently storing and accessing large amounts of data is difficult.
 - The additional demands of fault tolerance and backups makes things even more complicated.
- ▶ Manipulating large data sets involves running immensely parallel processes.
 - Gracefully recovering from any failures during such a run and providing results in a reasonably short period of time is complex.
- ▶ Managing the continuously evolving schema and metadata for semi-structured and un-structured data, generated by diverse sources, is a convoluted problem.

➤ So, New Data stores were needed

11

11

It started with Google



- ▶ At around 2000-2003 Google realised that they need a massively scalable infrastructure for their search engine and other applications.
 - Google's approach was to solve the problem at every level of the application stack.
- ▶ The goal was to build a scalable infrastructure for parallel processing of large amounts of data.
 - Google therefore created a full mechanism that included
 - a distributed filesystem,
 - a column-family-oriented data store,
 - a distributed coordination system,
 - and a MapReduce-based parallel algorithm execution environment.
- ▶ Over three years 2003-2006, Google, graciously enough, published and presented a series of papers explaining some of the key pieces of its infrastructure:
 - Google File System, 2003: <http://research.google.com/archive/gfs.html>
 - Chubby, 2006: <http://research.google.com/archive/chubby.html>
 - MapReduce, 2004: <http://research.google.com/archive/mapreduce.html>
 - Big Data, 2006: <http://research.google.com/archive/bigtable.html>

12

12

SQL? No, -> NoSQL

- ▶ No + SQL
- ▶ The creators of the buzzword *NoSQL* probably wanted to say *No RDBMS* or *No relational*
- ▶ A few others proposed that NoSQL is actually “Not Only SQL.”
- ▶ NoSQL is not a single product or even a single technology
 - It represents a class of products and a collection of diverse, and sometimes related, concepts about data storage and manipulation.

13

13

What is NoSQL?

- ▶ It's more than rows in tables
- ▶ It's free of joins
- ▶ It's schema-free (schemaless)
- ▶ It works on many processors
- ▶ It supports linear scalability
- ▶ It is innovative



14

14

What NoSQL is not:

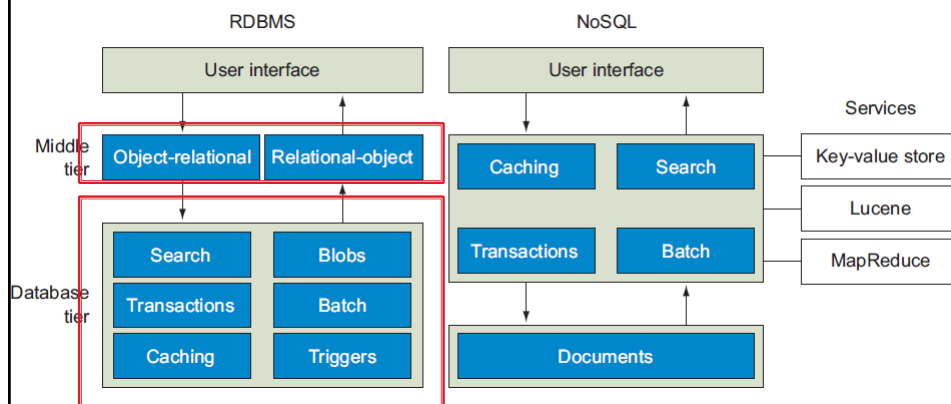


- ▶ It's not about the SQL language
 - SQL as well as other languages are used
- ▶ It's not only open source
 - Commercial products use NoSQL as well as open source initiatives
- ▶ It's not only big data
 - Though volume and velocity are important, NoSQL also focuses on variability and agility
- ▶ It's not about cloud computing
 - NoSQL solutions can run in the cloud as well as in your corporate data centre
- ▶ It's not an elite group of products
 - There is no membership

15

15

Application tiers



Making Sense of NoSQL

16

16

Part 2

17

17

RDBMS pros



- ▶ ACID transactions at the database level makes development easier.
- ▶ Fine-grained security on columns and rows using views prevents views and changes by unauthorized users.
- ▶ Most SQL code is portable to other SQL databases, including open source options.
- ▶ Typed columns and constraints will validate data before it's added to the database and increase data quality.
- ▶ Existing staff members are already familiar with entity-relational design and SQL.

Making Sense of NoSQL

18

18

RDBMS cons:



- ▶ The object-relational mapping layer can be complex.
- ▶ Entity-relationship modeling must be completed before testing begins, which slows development.
- ▶ RDBMSs don't scale out when joins are required.
- ▶ Sharding over many servers can be done but requires application code and will be operationally inefficient.
- ▶ Full-text search requires third-party tools.
- ▶ It can be difficult to store high-variability data in tables.

Making Sense of NoSQL

19

19

NoSQL Advantages



- ▶ High scalability
 - NoSQL can handle huge amount of data because of scalability
 - NoSQL databases use sharding for horizontal scaling.
 - Partitioning of data and placing it on multiple machines in a such way that the order of the data is preserved is sharding.
 - Vertical scaling means adding more resources to the existing machines whereas horizontal scaling means adding more machines to handle the data.
 - Horizontal scaling is easy to implement.
 - As the data grows NoSQL scale itself to handle that data in efficient manner.
- ▶ High availability
 - Auto replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.

20

20

NoSQL Advantages – continued

- ▶ Loading test data can be done with drag-and-drop tools before ER modeling is complete.
- ▶ Modular architecture allows components to be exchanged.
- ▶ Linear scaling takes place as new processing nodes are added to the cluster.
- ▶ Lower operational costs are obtained by autosharding.
- ▶ Integrated search functions provide high-quality ranked search results.
- ▶ There's no need for an object-relational mapping layer.
- ▶ It's easy to store high-variability data.



Making Sense of NoSQL

21

21

NoSQL Disadvantages



- ▶ Narrow focus:
 - NoSQL databases have very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are better choice in the field of Transaction Management than NoSQL
- ▶ Open-source:
 - NoSQL is open source databases. There is no reliable standard for NoSQL yet.
- ▶ Management challenge:
 - The purpose of big data tools is to make management of a large amount of data as simple as possible, but it is not easy with NoSQL

22

22

NoSQL Disadvantages – continued

- ▶ ACID transactions not always supported.
- ▶ Document stores don't provide fine-grained security at the element level.
- ▶ NoSQL systems are new to many staff members and additional training may be required.
- ▶ The document store has its own proprietary nonstandard query language, which prohibits portability.
- ▶ The document store won't work with existing reporting and OLAP tools.



Making Sense of NoSQL

23

23

Part 3

ACID and BASE

24

24

Consistency

- ▶ Consistency can be defined by how the copies from the same data may vary within the same replicated database system.
- ▶ For many years, system architects would not compromise when it came to storing data and retrieving it.
- ▶ Strong consistency was not a choice. It was a requirement for all systems.



25

25

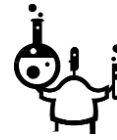
Consistency models

- ▶ The two most common consistency models are known by the acronyms ACID and BASE
- ▶ **ACID:**
 - **Atomic**
 - All operations in a transaction succeed or all operations are rolled back.
 - **Consistent**
 - On the completion of a transaction, the database is structurally sound and left in a consistent state.
 - **Isolated**
 - Transactions do not contend with one another. Continuous access to data is moderated by the database so that transactions appear to run sequentially.
 - **Durable**
 - The results of applying a transaction are permanent, even in the presence of failures

26

26

ACID



- ▶ ACID properties mean that once a transaction is complete, its data is consistent (the term used often - **write consistency**) and stable on disk, which may involve multiple distinct memory locations.
- ▶ Write consistency is a very useful mechanism for application developers, but it also requires sophisticated locking which is typically a heavyweight pattern for most use cases.

27

27

BASE

- ▶ For many NoSQL datastores ACID consistency model is too restrictive, as they have other priorities like scale and resilience.
- ▶ **BASE**:¹
 - **B**asic **A**vailability
 - The database appears to work most of the time.
 - **S**oft state
 - Stores don't have to be write-consistent, nor do different replicas have to be mutually consistent all the time.
 - **E**ventual consistency
 - Stores exhibit consistency at some later point (e.g., lazily at read time).

1. The term was coined by Eric Brewer

28

28

BASE vs ACID

- ▶ BASE properties are much looser than ACID, but there isn't a direct one-for-one mapping between the two consistency models.
- ▶ A BASE data store values availability (since that's important for scale), but it doesn't offer guaranteed consistency of replicated data at write time.
- ▶ Overall, the BASE consistency model provides a less strict assurance than ACID: data will be consistent in the future.
- ▶ A fully ACID database is the perfect fit for use cases where data reliability and consistency are essential, for example, banking.
- ▶ However there are some NoSQL data stores that support ACID
 - CouchDB and Neo4j are two examples of NoSQL databases that provide strong consistency and are ACID compliant.

29

29

Brewer's CAP theorem

- ▶ Eric Brewer in 2000 introduced the CAP theorem that states:
- ▶ **It is impossible** for a distributed computer system to simultaneously provide
 - **C**onsistency,
 - **A**vailability,
 - **P**artition tolerance
- ▶ Maximum **two** of the three can be provided at any given point in time.

G.Vaish (2013) *Getting Started with NoSQL*, Packt Publishing

30

30

Consistency

- ▶ Having a single, up-to-date, readable version of your data available to all clients.
- ▶ Consistency here is concerned with multiple clients reading the same items from replicated partitions and getting consistent results.



31

31

High Availability

- ▶ Knowing that the distributed database will always allow database clients to update items without delay.
- ▶ Internal communication failures between replicated data shouldn't prevent updates.

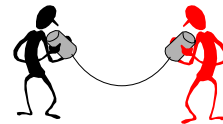


32

32

Partition tolerance

- ▶ The ability of the system to keep responding to client requests even if there's a communication failure between database partitions.
 - This is analogous to a person still having an intelligent conversation even after a link between parts of their brain isn't working.
- ▶ Partition tolerance is strictly an architectural decision (will the database be distributed or not).
 - Distributed databases must be partition tolerant



33

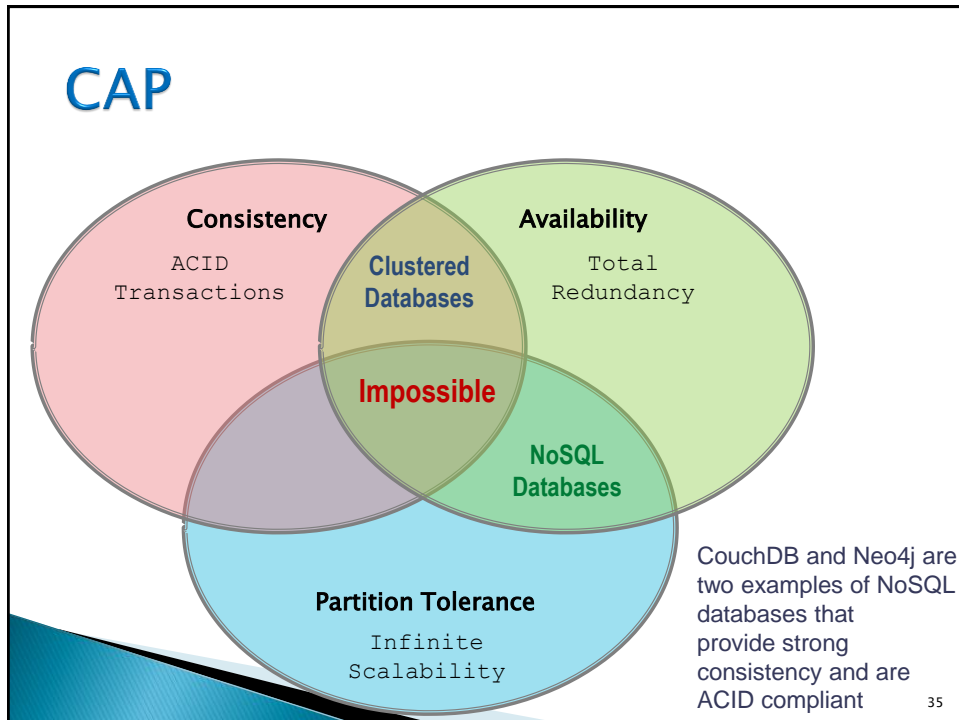
33

CAP details

- ▶ CAP proves that you can create a distributed database that is
 - **consistent** (writes are atomic and all subsequent requests retrieve the new value),
 - **available** (the database will always return a value as long as a single server is running), or
 - **partition tolerant** (the system will still function even if server communication is temporarily lost—that is, a network partition),
- ▶ but you can have **only two at once**:
 - *It is not possible to create a distributed database that is consistent and available and partition tolerant at the same time.*

34


34



35

Considerations when choosing

- ▶ What does my data look like?
 - Choose the right type of the database
- ▶ How is the current data stored?
 - Do you need to migrate data or it is possible to keep legacy in RDBMS and move forward with new data in a NoSQL data store?
- ▶ How important is the guaranteed accuracy of database transactions?
 - If yes, stay with RDBMS.
- ▶ How important is the speed of the database?
 - Many NoSQL stores can provide a huge performance boost.
- ▶ What happens when the data is not available?
 - Many NoSQL, as well as RDBMS solutions provide a good high availability.
- ▶ How is the database being used?
 - Are the most operations on the database *writes* to store data, or they are *reads* to get data?



36

Further reading

- ▶ Singh, A., Sultan A.. Data Modeling With NoSQL Database. Independently published, 2019
- ▶ Gaspar, D. Bridging relational and NoSQL databases. Hershey, PA, USA : IGI Global, 2018
- ▶ Harrison, G. Next generation databases : NoSQL, NewSQL, and Big Data. Apress, 2015
- ▶ CAP-based Examination of Popular NoSQL Database Technologies in Streaming Data Processing. 2019 International Artificial Intelligence and Data Processing Symposium (IDAP) Artificial Intelligence and Data Processing Symposium (IDAP), 2019 International. :1-6 Sep, 2019
- ▶ Hendricks, W. Review of NoSQL Data Stores: Using a reactive three-tier application for software developers to achieve a high availability application design architecture,, 2019 Open Innovations (OI) Open Innovations (OI), 2019. :71-77 Oct, 2019

37

37

Essentials

- ✓ Discussed challenges of RDMS
- ✓ Introduced NoSQL
- ✓ Compared RDMS and NoSQL
- ✓ Discussed BASE vs ACID
- ✓ Introduced CAP theorem

38

38