

Deciphering Big Data Module - Individual Project:

Executive Summary

Big Data Summary

Data back in the 1970s was very simple and basic, with the concept being to store structured data, such as variables and strings. From then, the introduction of newer technologies and the World Wide Web, lead to the term 'Big Data' becoming used more frequently, after reflections on the need to handle data beyond traditional database capabilities. Especially with the use of unstructured data such as videos.

Big Data refers to exceptionally large datasets that are complex and difficult to process using traditional data processing applications. It includes the collection, storage, and analysis of data from a variety of sources, such as social media, online transactions, and databases, which are used for data acquisition and storage. The characteristics of big data can be defined by the four V's: Velocity, Veracity, Volume and Variety (Černiauskas 2022). The four V's refers to the sheer volume of data generated and collected, how fast the data is moved, the variety of different data types, and the trustworthiness of the data.

Group Project Overview and Database Design (SQL & NoSQL)

For our group project, we were advised to position ourselves as IT Software Consultants and Developers. In which, we were commissioned to design and build a single logical database. We decided on creating a database management system for a school. The objective of the school system is to manage and monitor data concerning students and their academic progress, including class attendance and assignment results. Student information would then be collected from enrolment forms, with regular requests for updates from parents to ensure all information is up to date.

We discussed ideas on methods to storing the data, and it meant deciding on using either SQL (Structured Query Language), or NoSQL (Non-Structured Query Language). Either would be suitable for our school system. However, 'there are always trade-offs among the set of quality attributes, and a software architect faces the difficult task of balancing them' (Khan et al 2023). In Table 1, the advantages and disadvantages of SQL and NoSQL databases have been outlined, showing some of the trade-offs faced for digital architects and data scientists. For example, whilst NoSQL is able to handle unstructured data, it loses the ability to create complex queries, unlike SQL which follows ACID properties for reliable processing.

With the current specification discussed with my fellow peers, we weighed the trade-offs between SQL and NoSQL databases and determined that SQL would be more appropriate for the types of data we plan to collect. We considered factors such as data structure, consistency, scalability, and the complexity of queries we anticipate. Given that our data will be highly structured and relational, requiring complex queries and transactional integrity, the robust nature of SQL databases seemed ideal. MySQL emerged as the preferred relational database management system (RDMS) of choice due to its reliability, widespread adoption, strong community support, and extensive documentation. Additionally, MySQL offers a balance of performance, ease of use, and scalability that aligns well with our project's needs, ensuring efficient data management and retrieval as we scale.

Table 1, Advantages and Disadvantages of SQL and NoSQL Databases:

	Advantages	Disadvantages
SQL (e.g. MySQL, PostgreSQL)	<ul style="list-style-type: none"> • Structured Datasets – structured predefined schemas for consistency • Robustness – standardised language to handle large complex queries and data manipulation. • Community and Support – extensive documentation and communities to bring professional support. • ACID Compliance - follow <u>ACID</u> (Atomicity, Consistency, Isolation, Durability) properties, which ensure reliable transaction processing. 	<ul style="list-style-type: none"> • Scalability – Lack of scalability horizontally for SQL databases. Hard to distribute storage/processing power to other machines (might not be able to scale up current machine if maxed out). • Cost - Licensing and operational costs can be high, when scaling vertically and for operational licenses. • Normalization and duplicated data – Creating too many tables and joins/relationships can slow down query performance and redundant data.
NoSQL (e.g. MongoDB, Riak)	<ul style="list-style-type: none"> • More Scalability – Unlike SQL, scalability vertically and horizontally is much easier with NoSQL. Cost advantage to business, with high availability and fault tolerance. • Flexibility of Data Types – Can handle unstructured, semi-structured, and structured data (e.g. videos, csv, excel). • Schema-less design – Ease of use advantage when modifying to reduce complexity. 	<ul style="list-style-type: none"> • Complex queries – Difficult to complete very complex queries with multiple joins efficiently. • Backup and Recovery – use of different data types and reduction in data integrity can make backup/recovery more complex. • ACID not required – Files don't have to follow ACID properties, so reliability in transaction processing may be absent.

MySQL Code and Database Creation

In the first assignment with my peers, we designed an initial layout for our RDMS. This involved determining the tables we wanted to include in our database and specifying the primary and foreign keys for these tables to establish the necessary joins and relationships, as detailed in Appendix A. Using MySQL, we leveraged the 'CREATE TABLE' function to set up our tables and the 'INSERT' command to populate them with dummy data, as demonstrated in Appendix B. This process included defining the appropriate data types for each column, such as 'VARCHAR(50)' for text fields.

Once the tables were created, I could then craft complex queries and joins to generate meaningful outputs. For instance, one of our queries provided a list of students along with the subjects they were studying, and the teachers assigned to each subject class, as shown in Appendix C. This demonstrated the powerful querying capabilities of MySQL and other SQL RDMS, particularly in handling intricate data relationships and producing detailed reports.

One of the most valuable features of MySQL is the ability to create custom queries using the 'WHERE' clause. In Appendix D, I needed to check the academic progress of pupils by identifying those who had missed classes. By using the 'WHERE' clause, I was able to filter the data to display only those students who had not attended class. This query revealed that pupils 'Chris' and 'Dave' missed a class on the 25th of April 2024. This capability highlights the strength of MySQL in performing specific and detailed data analysis, enabling us to monitor and manage student attendance effectively.

Future Considerations and NoSQL Databases

As mentioned by Khan et al (2023), the needs of an individual/organisation will determine which is more appropriate to use, SQL or NoSQL. While MySQL is capable of meeting the basic needs of a school in tracking academic progress and providing layered security between pupils, parents, and teachers, there are future considerations that must be taken into account. The increasing diversity and volume of data necessitate a revaluation of our database strategy, indicating that NoSQL might become more appropriate in the long term.

Building a NoSQL database foundation offers numerous opportunities for future expansion. The inherent flexibility and scalability of NoSQL databases support both horizontal and vertical scaling, which is crucial for evolving into a large-scale business model. This adaptability would enable us to offer our system to multiple schools or even secure government contracts, significantly broadening a potential target market. Moreover, the big data capabilities of NoSQL databases allow for comprehensive data analysis, facilitating the observation of trends and patterns across different regions. This deeper analytical insight could be invaluable for educational policymaking and improving school performance.

A NoSQL choice for the future would be MongoDB due to several compelling reasons discussed above. Unlike traditional SQL databases, MongoDB stores data in flexible, JSON-like documents, allowing for varied data structures within a single collection. This adaptability makes it easier to modify data schemas as requirements evolve without the need for complex migrations, which is particularly beneficial for the educational sector, which is seeing rapid change and development in the way teaching methods are being applied.

With digital education evolving rapidly and allowing students to access educational content 24/7. Traditional relational databases are inefficient for handling large volumes of data like internet, multimedia, and social media (Györödi et al 2015). For instance, students in this course need access to lecture recordings. MongoDB's GridFS feature simplifies storing and retrieving video content, making it easier for students to find relevant materials for their learning.

Overall, transitioning to a NoSQL database would give you the ability to handle current data demands whilst future proofing for potential infrastructure growth and advanced data analytics. A NoSQL database management system like MongoDB, would provide a flexible schema, scalability, efficient handling of large files, advanced querying capabilities, and ease of integration with big data tools, making it an ideal choice for managing school data and video files.

General Data Protection Regulation (GDPR) considerations for SQL and NoSQL

When developing databases, ensuring the databases complies with GDPR is top priority for protecting personal data privacy and maintaining legal standards. Managing children's personal data is especially sensitive, and GDPR mandates responsible handling, which includes minimising data collection, obtaining user consent, and respecting individuals' rights to access, rectify, and erase their data. Adhering to GDPR protocols means collecting only the essential data, thus reducing potential risks. Implementing strong security measures, such as encryption and pseudonymisation, is vital to prevent unauthorized access and data breaches, in both SQL and NoSQL systems. As highlighted by the European Commission (2016), non-compliance can result in hefty fines and reputational damage. Prioritizing GDPR compliance not only helps avoid these penalties but also builds trust with users, demonstrating a commitment to ethical procedures in managing sensitive and personal data.

Conclusion

SQL databases such as MySQL were appropriate for creating a simple school database due to their rigid schema and the lack of need to manage semi-structured or unstructured data for tracking pupil performance. The structured nature of SQL databases ensures data consistency and integrity, which is essential for the straightforward requirements of managing student records and academic progress. However, the landscape of digital education is rapidly evolving, and the expansion of Big Data necessitates a forward-thinking approach to database design. As educational institutions increasingly rely on diverse data sources, including multimedia content and real-time analytics, the limitations of traditional SQL databases become more apparent.

NoSQL databases provide users with flexible schemas, scalability, efficient handling of large files, advanced querying capabilities, and seamless integration with big data tools. For governments like the UK, implementing a unified NoSQL database system to manage pupil data nationwide could eliminate the need for individual schools to develop and maintain their own databases. This centralised approach would standardise data management procedures and user interfaces across all schools, enhancing consistency and ease of use. Additionally, it would empower students to enhance their learning by analysing unstructured data such as videos and lecture recordings. The cost-effectiveness of this solution is particularly relevant given the financial constraints on schools, with total school spending per pupil in England falling by 9% in real terms between 2009–10 and 2019–20 (Institute of Fiscal Studies, 2024). A unified NoSQL system would not only streamline operations but also offer significant economic benefits by reducing redundancy and leveraging economies of scale.

References –

European Commission (2016) GDPR Regulation of the European Parliament and Council. Available at: <http://data.europa.eu/eli/reg/2016/679/oj> [Accessed: 20th May 2024].

Khan W, Kumar T, Zhang C, Raj K, Roy AM, Luo B (2023) SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review. Available at: <https://www.mdpi.com/2504-2289/7/2/97> [Accessed: 23rd May 2024].

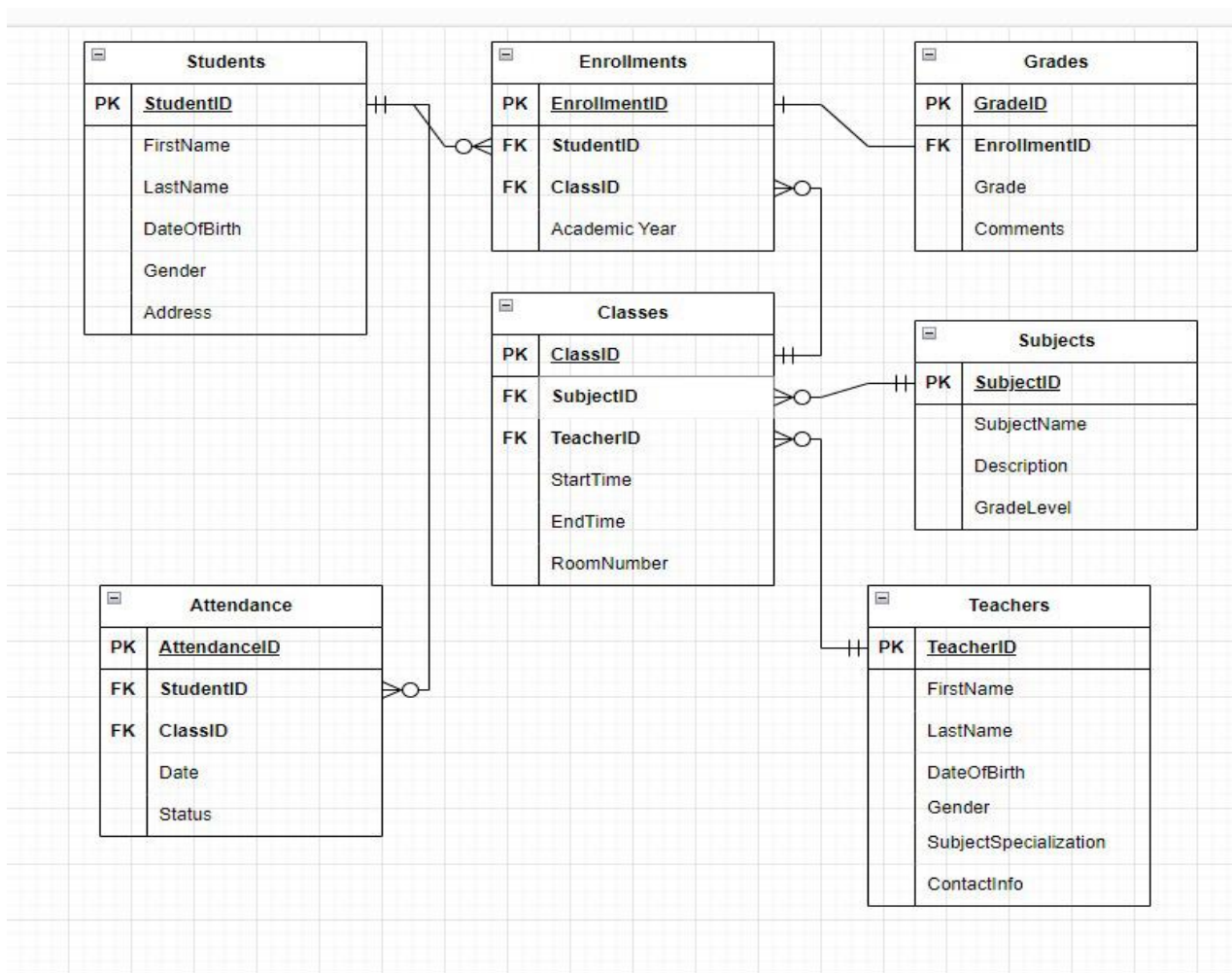
Černiauskas J (2022) Understanding The 4 V's Of Big Data, Forbes Technology Council. Available at: <https://www.forbes.com/sites/forbestechcouncil/2022/08/23/understanding-the-4-vs-of-big-data> [Accessed: 23rd May 2024].

Győrödi C, Győrödi R, Pecherle G, Olah A (2015) A comparative study: MongoDB vs. MySQL, 13th International Conference on Engineering of Modern Electric Systems (EMES). Available at: <https://doi.org/10.1109/EMES.2015.7158433> [Accessed: 24th May 2024].

Institute of Fiscal Studies (2024) Education spending Spring Budget 2024, The latest picture on school funding and costs in England, Available at: <https://ifs.org.uk/articles/latest-picture-school-funding-and-costs-england> [Accessed: 24th May 2024].

Appendices –

Appendix A, Database management system structure



Appendix B, Creating Tables and inserting dummy data

#Creating a student table, with fictional data

```
CREATE TABLE students (
    student_id INT NOT NULL,
    firstname VARCHAR(50),
    lastname VARCHAR(50),
    dateofbirth DATE,
    gender VARCHAR(50),
    address VARCHAR(50)
);
```

#Creating our 4 fictional students

```
INSERT INTO students (student_id,firstname,lastname,dateofbirth,gender,address) VALUES
(1,'Chris','Final','2000-02-15','Male','12 Dixon Close, Braintree'),
(2,'Dave','Smith','2001-10-01','Male','125A Broadway, Witham'),
(3,'Jenny','Jones','2000-01-11','Female','14 Notlet Rd, Braintree'),
(4,'Saoirse','Lee','2001-03-25','Male','171 Braintree Rd, Witham');
SELECT * FROM students; #Checking our data has been created.
```

Appendix C, Creating complex queries/joins for displaying students and subject teachers

```
96 #Creating a complex queries of joins to display students names and their subjects/teachers
97 • SELECT students.firstname AS 'Student FN',
98         students.lastname AS 'Student LN',
99         students.gender,
100         subjects.subjectname,
101         teachers.firstname AS 'Teacher FN',
102         teachers.lastname AS 'Teacher LN'
103 FROM students
104 JOIN enrolments ON students.student_ID=enrolments.studentid
105 JOIN classes ON enrolments.classid=classes.class_id
106 JOIN subjects ON classes.subjectid=subjects.subject_id
107 JOIN teachers ON classes.teacherid=teachers.teacher_id;
108
```

Result Grid

Filter Rows:



Export:

Wrap Cell Content:

	Student FN	Student LN	gender	subjectname	Teacher FN	Teacher LN
▶	Chris	Final	Male	Data Science	Jeremy	Clarkson
	Dave	Smith	Male	Data Science	Jeremy	Clarkson
	Jenny	Jones	Female	Maths	Claire	Ocean
	Saoirse	Lee	Male	Maths	Claire	Ocean

Appendix D, creating complex queries to track academic performance (e.g. Missed Attendance)

```
109  #Queries with WHERE commands to find academic performance (e.g. Class attendance)
110  •  SELECT  students.firstname,
111         attendance.Classdate,
112         attendance.attendanceflag
113  FROM  students
114  JOIN  enrolments ON students.student_ID=enrolments.studentid
115  JOIN  classes ON enrolments.classid=classes.class_id
116  JOIN  attendance ON attendance.classid=classes.class_id
117  WHERE attendance.attendanceflag=0; #WHERE command showing Chris and Dave missed class on 25th April 2024
```

Result Grid |   Filter Rows: | Export:  | Wrap Cell Content: 

	firstname	Classdate	attendanceflag
▶	Chris	2024-04-25	0
	Dave	2024-04-25	0