**Name:** Julio Pochet Edmead  
**Assignment Title:** Module 1.3 Assignment: Basic Comparison of Relational vs. NoSQL Databases  
**Date:** 01/12/2025

**Relational vs. NoSQL Databases**

**What Are Relationships in Relational Databases?**

In relational databases, relationships define how data in one table connects to data in another. These relationships make it easier to structure and retrieve information in a meaningful way. Here are two common types of relationships:

* **One-to-One (1:1):**  
  This means that one record in a table is linked to one record in another table. For example, a user might have one profile, and that profile is tied to only one user. This type of relationship is great for storing sensitive or additional information separately, like a user and their account settings.
* **One-to-Many (1:N):**  
  This is when one record in a table links to multiple records in another. Think of a customer placing orders at an online store: one customer can have multiple orders, but each order is tied to just one customer. This is one of the most common relationship types in relational databases.

Relationships like these are built using **primary keys** (unique identifiers in a table) and **foreign keys** (references to those identifiers in another table). These connections ensure the database stays organized and consistent.

**Advantages of Relational Databases and NoSQL Databases**

* **Relational Databases:**
  + They are reliable because of their strict structure, making them ideal for systems where consistency is necessary, like banking.
  + ACID compliance (Atomicity, Consistency, Isolation, Durability) ensures data accuracy even in complex transactions.
  + SQL, the standard query language, is widely used and supported.
* **NoSQL Databases:**
  + They easily handle unstructured or semi-structured data, like social media posts or IoT sensor data.
  + These databases are built for scalability. If your data grows rapidly, adding more servers (horizontal scaling) is simple and efficient.
  + NoSQL is perfect for real-time applications, like gaming leaderboards or stock market tracking.

**Disadvantages of Relational Databases and NoSQL Databases**

* **Relational Databases:**
  + Their rigid schema can be a headache when data requirements change. You’ll often need to redesign tables and relationships, which takes time.
  + They don’t scale well horizontally, limiting performance when handling large datasets.
* **NoSQL Databases:**
  + Many NoSQL systems trade-off strict consistency for scalability, which might not work for applications needing real-time precision.
  + They lack the standardization of relational databases, making complex queries or migrations between systems more challenging.
  + Some NoSQL databases don’t provide as much built-in support for data integrity.

**Features of MySQL and MongoDB**

* **MySQL Features:**
  1. **ACID Compliance:**  
     MySQL is known for its reliability during transactions, ensuring that banking or e-commerce platforms don’t lose data. For example, if you’re buying something online, MySQL ensures your payment is processed correctly without duplicating or losing records.
  2. **Joins:**  
     This feature lets you pull data from multiple related tables in a single query. For example, you can get customer details and their orders in one go.
* **MongoDB Features:**
  1. **Flexible Schema:**  
     MongoDB stores data as JSON-like documents. This makes adapting when your data structure changes easier, perfect for a startup adding new features.
  2. **Horizontal Scalability:**  
     MongoDB can handle growing data by distributing it across multiple servers (sharding). For instance, a company like Netflix might use this feature to seamlessly manage its massive library of movies and user data.

**Conclusion**

Relational databases and NoSQL databases each have strengths and weaknesses. Relational databases shine in structured environments where consistency is key, while NoSQL databases dominate in scenarios requiring flexibility and scalability. Choosing the right one depends on your specific needs, but understanding their features and trade-offs is essential for designing efficient systems.

**References**

MongoDB Inc. *What is MongoDB?* <https://www.mongodb.com>  
MySQL Documentation. *What is MySQL?* <https://dev.mysql.com/doc>