**Comparison between Relational and Non-Relational Database**

|  |  |  |
| --- | --- | --- |
|  | Relational Database | Non-Relational Database |
| DB Schema | * Structured Query Language * Stores items in tables to minimize duplicate values * Fixed schema * Example: MySQL, Oracle | * NoSQL database * Unstructured data * No fixed schema * Document-oriented * Example: MongoDB, HBase |
| Scalability | * The tables relationship makes scaling more resource-intensive | * Stores each item as single document for high scalability * Includes sharding or partitioning * Uses elastic scalability |
| Flexibility | * Provides flexible structure to meet changing requirements and increasing amounts of data * This model permits changes to a database structure to be implemented easily without impacting the data or the rest of the database * In reality, growth and change are limited by the relational database management system and physical computing hardware | * Flexible data model * Easy to store and combine data of any structure * Defining types of data in advance is not required * Allows dynamic modification of schema without performance impact |
| Cost | * Rely on expensive proprietary servers and storage systems * Licenses for this system can be quite expensive | * Uses clusters of cheap commodity servers * Databases are open source and therefore free |
| consistency | * Tight consistency | * Eventual consistency rather than ACID property |
| Transaction | * Transaction with ACID property (Atomicity, Consistency, Isolation & Durability) | * Does not support transactions |
| Performance | * High performance speed | * High read and write performance * Unlimited growth with higher throughput * Lower latency than relational database * Faster development life cycles for developers |
| Reliability | * Changes committed in a transaction are stored and available in the database even if there is power failure or the database goes offline suddenly. | * Automatic back up of data in separate facilities (Eg. In DynamoDB) |
| Reporting tools | * Wide array of reporting tools helps to prove application’s validity | * Lack of reporting tools for analysis and performance testing |
| Security [2] | * By splitting data into tables, certain tables can be made confidential * The system can then limit access only to those tables whose records they are authorised to view | * Has weak password storage * Lack of encryption support for the data files * Weak authentication both between client and the servers * Vulnerability to SQL injection * Denial of service attacks. |

[2][1][3][4][5][6][7]