MT:

a)

1.Definitions

Relational database: Refers to a database that uses a relational model to organize data.

NoSQL database: The full name is Not Only SQL, which refers to non-relational databases. It's a way to span multiple database types

2. Benefits

Relational database:

Data structuring, data integrity, clear tabular relationships, support SQL language, efficient data query, support ACID transaction management, multi-level security mechanism, more flexible data operation, easy to maintain and expand, widely used and compatible

NoSQL database:

It supports a variety of data models, high scalability, high performance, high concurrency, distributed architecture, low latency, easy to process big data, adapts to dynamic data, establishes an active developer community and rich ecosystem, and provides diversified tools and support

3.Limitations

Relational database:

Lack of flexibility, limited scalability, performance bottlenecks, insufficient high-concurrency processing capacity, and complex management and maintenance

NoSQL database:

There are consistency issues, limited query capabilities, low standardization, insufficient transaction support, and data redundancy and complexity

4.Example:

Relationship database:

MySQL, PostgreSQL , Microsoft SQL Server, Oracle Database, SQLite

NoSQL database;

Mongo DB, Cassandra, Redis, Couchbase, Neo4j, DynamoDB

5.Use Cases

Relational database:

Financial Applications, Enterprise Resource Planning, Customer Relationship Management, Content Management Systems, E-commerce Platforms

NoSQL database;

Big Data Applications, Real-time Analytics, Content Delivery Networks, Gaming Applications, Social Networks, IoT Applications

b):

Feature	Relational Databases	NoSQL Databases
Examples	MySQL, Microsoft SQL Server	Redis, Memcached

Data Structured	Structured Data	Unstructured Data
Data Storage	Small	Large
ACID transactions	Support	Not support or limited. Some support consistency.
Normalisation supported	Support	Not support
Integrity constraints	Support	Limited
Scalability	Vertical	Horizontal
Simplicity	Complex. Support large available	Easy. Support small available
Complexity Cost	Higher	Lower
Reliability	High	Low
Schema Flexibility	Low	High
Performance	Optimized for complex queries; may struggle with large datasets	High performance, especially for large datasets
Storage Requirements	Requires more storage for metadata and indexes	More efficient storage, can store large amounts of unstructured data