Understanding the difference between SQL and NoSQL

	SQL	NoSQL
Data storage	Stored in a relational model, with rows and columns. Rows contain all of the information about one specific entry/entity, and columns are all the separate data points; for example, you might have a row about a specific car, in which the columns are 'Make', 'Model', 'Color' and so on.	The term "NoSQL" encompasses a host of databases, each with different data storage models. The main ones are: document, graph, key-value and columnar. More on the distinctions between them below.
Schemas and Flexibility	Each record conforms to fixed schema, meaning the columns must be decided and locked before data entry and each row must contain data for each column. This can be amended, but it involves altering the whole database and going offline.	Schemas are dynamic. Information can be added on the fly, and each 'row' (or equivalent) doesn't have to contain data for each 'column'.
Scalability	Scaling is vertical. In essence, more data means a bigger server, which can get very expensive. It is possible to scale an RDBMS across multiple servers, but this is a difficult and time-consuming process.	Scaling is horizontal, meaning across servers. These multiple servers can be cheap commodity hardware or cloud instances, making it a lot more cost-effective than vertical scaling. Many NoSQL technologies also distribute data across servers automatically.
ACID Compliancy (Atomicity, Consistency, Isolation, Durability)	The vast majority of relational databases are ACID compliant.	Varies between technologies, but many NoSQL solutions sacrifice ACID compliancy for performance and scalability

Initially data was stored in files. However, as the amount of data increased, it was not convenient to access the data using files. It was a slow and inefficient process. As the amount of data grew, it was very difficult to maintain the data and fetch any record. Hierarchical and Network databases were designed as storage mechanisms but they did not provide a standard method to access the data.

With the need to manage data and the desire for a standard method to access data, SQL came into existence

RDBMS: Relational Database

Main Focus of RDBMS is on ACID properties:

- Atomicity – Each transaction is atomic. If one part of it fails, the entire transaction fails (and is rolled back)
- Consistency – Every transaction is subject to a consistent set of rules (constraints, triggers, cascades)
- Isolation – No transaction should interfere with another transaction
- Durability – Once a transaction is committed, it remains committed

- ACID is important
  - But only when it's important
  - i.e. banking, finance, safety systems, etc.
  - The kinds of systems that people were building with computers 30 years ago (and today)
- ACID adds overhead
  - Features like atomicity, isolation basically force database servers to use sequential evaluation

The CAP theorem:

- Impossible for any shared data-system to guarantee simultaneously all of the following three properties:
  - Consistency – once data is written, all future read requests will contain that data
  - Availability – the database is always available and responsive
  - Partition Tolerance – if part of the database is unavailable, other parts are unaffected

NoSQL

- RDBMS doesn't quite fit for *some* requirements
- Google and Amazon decided to make their own stuff (BigTable and S3 Storage) to meet their own unique needs
    - Distributed
    - Scalability
    - Control over performance characteristics
    - High availability
    - Low Latency
    - Cheap

What is NoSQL?

- Basically a large serialized object store
    - (mostly) retrieve objects by defined ID
- In general, doesn't support complicated queries
- Doesn't have a structured (or any!) schema
    - Recommends denormalization
- Designed to be distributed (cloud-scale) out of the box
- Because of this, drops the ACID requirements
    - Any database can answer any query
    - Any write query can operate against any database and will "eventually" propagate to other distributed servers

* Dependent on vendor

The Opposite of ACID

- BASE
    - Basically Available – guaranteed availability

    - Soft-state – the state of the system may change, even without a query (because of node updates)
    - Eventually Consistent – the system will become consistent over time
- Contrived acronym, but so is ACID :P

NoSQL-Eventual Consistency

- Because of the distributed model, any server can answer any query
- Servers communicate amongst themselves at their own pace (behind the scenes)
- The server that answers your query might not have the latest data
- Who really cares if you see Kim Kardashian's latest tweet instantaneously?

Different Types of NoSQL

- Column Store
    - Column data is saved together, as opposed to row data
    - Super useful for data analytics
    - Hadoop, Cassandra, Hypertable
- Key-Value Store
    - A key that refers to a payload
    - MemcacheDB, Azure Table Storage, Redis
- Document / XML / Object Store
    - Key (and possibly other indexes) point at a serialized object
    - DB can operate against values in document
    - MongoDB, CouchDB, RavenDB
- Graph Store
    - Nodes are stored independently, and the relationship between nodes (edges) are stored with data

Conclusion

Limitations for SQL database

Scalability: Users have to scale relational database on powerful servers that are expensive and difficult to handle. To scale relational database it has to be distributed on to multiple servers. Handling tables across different servers is a chaos.

Complexity: In SQL server's data has to fit into tables anyhow. If your data doesn't fit into tables, then you need to design your database structure that will be complex and again difficult to handle.

- RDBMS is a great tool for solving ACID problems
    - When data validity is crucial
    - When you need to support dynamic queries
- NoSQL is a great tool for solving data availability problems
    - When it's more important to have fast data than up-to-the-minute just updated data
    - When you need to scale based on changing requirements
- Pick the right tool for the job

References:

RTigger power point presentations

http://dataconomy.com/