NoSQL Data Models

**When Not to Use SQL:**

* **Need high Availability in the data**: Indicates the system is always up and there is no downtime
* **Have Large Amounts of Data**
* **Need Linear Scalability**: The need to add more nodes to the system so performance will increase linearly
* **Low Latency**: Shorter delay before the data is transferred once the instruction for the transfer has been received.
* **Need fast reads and write**

In a distributed database, in order to have **high availability**, you will need copies of your data.

### Eventual Consistency:

Over time (if no new changes are made) each copy of the data will be the same, but if there are new changes, the data may be different in different locations. The data may be inconsistent for only milliseconds. There are workarounds in place to prevent getting stale data.

**CAP Theorem:**

* **Consistency**: Every read from the database gets the latest (and correct) piece of data or an error
* **Availability**: Every request is received and a response is given -- without a guarantee that the data is the latest update
* **Partition Tolerance**: The system continues to work regardless of losing network connectivity between nodes

**Data Modeling in Apache Cassandra:**

* Denormalization is not just okay -- it's a must
* Denormalization must be done for fast reads
* Apache Cassandra has been optimized for fast writes
* ALWAYS think Queries first
* One table per query is a great strategy
* Apache Cassandra does **not** allow for JOINs between tables