1. **O que é Big Data?**

Big Data é um termo para descrever um grande volume de dados, seja ele estruturado ou não, sua importância não está ligado apenas a quantidade de dados mas o que é feito com eles, normalmente esses dados são usados para gerar as melhores decisões e estratégias de negócios.

O conceito Big Data pode ser novo para algumas pessoas porém a questão é, geramos muito mais dados hoje do que antigamente, o diferencial do Big Data está justamente atrelado à possibilidade e oportunidade em cruzar esses dados por meio de diversas fontes para obtermos insights rápidos e preciosos.

2. **O que é NoSQL?**

É fácil encontrar na internet algumas definições bem confusas que passam a ideia de um conceito que tenta acabar com o padrão SQL, porém o NoSQL não é nada disso, NoSQL é um termo usado para descrever bancos de dados não relacionais de alto desempenho. Os bancos de dados NoSQL usam diversos modelos de dados, incluindo documentos, gráficos, chave-valor e colunares.

Bancos de dados NoSQL são amplamente reconhecidos pela facilidade de desenvolvimento, desempenho escalável, alta disponibilidade e resiliência.

Umas das características mais importantes do NoSQL são:

- Utilização do processamento paralelo para processamento das informações.
- Distribuição em escala global.
- Banco de dados que trabalham no esquema (chave/valor).
- Banco de dados orientados a documentos.

Alguns bancos que utilizam esse modelo são:

- MongoDB
- Neo4J
- Cassandra
- etc...

3. **Qual a relação entre Big Data e NoSQL?**

Quando se trabalha com Big Data, estamos trabalhando com grandes quantidade de dados, com mais desempenho e mais esquemas flexíveis são os dois maiores condutores de NoSQL.

A necessidade de sistemas de bancos de dados mais eficientes para estas tarefas motivou a utilização dos bancos de dados NoSQL e os NewSQL.

Os bancos NoSQL tiveram um crescente aumento com a difusão dos Big Data, chegando algumas empresas a desenvolverem bancos internos para atender inicialmente demandas da própria empresa, onde acabaram se tornando um produto, como o caso do BigTable e Dynamo, desenvolvidos para solucionar problemas com larga escala que sofriam no momento.

4. **O que é NewSQL?**

São bancos de dados que buscam promover a mesma melhoria de desempenho e escalabilidade dos sistemas NoSQL, não abrindo mão dos benefícios dos bancos de dados tradicionais, da linguagem SQL e das propriedades ACID.

Suas principais características são:

- Linguagem SQL como meio de interação entre o SGBD e a aplicação
- Suporte para transações ACID
- Controle de concorrência não bloqueante, para que as leituras e escritas não causem conflitos entre si
- Arquitetura que fornece um maior desempenho por nó de processamento
- Arquitetura escalável, com memória distribuída e com capacidade de funcionar em um aglomerado com um grande número de nós

5. **Cite e explique o máximo de Vs relacionados à Big Data que você descobrir (no mínimo 5).**
    - Volume: Considerando que a quantidade de dados gerado diariamente já é maior que a quantidade acumulada de vários anos atrás.
    - Velocidade: Porque na maioria das vezes precisamos extrair informações rapidamente, como num sistema de tráfego.
    - Variedade: Muitos dos dados vem de redes sociais, e-mais, documentos entre outros dados não estruturados.
    - Veracidade: Precisamos ter certeza que os dados tem um sentido e que sejam autênticos.
    - Valor: É necessário que a organização obtenha retorno com o investimento.
    - Variabilidade: Podem ser encontradas anomalias ou resultados isolados.
    - Validade: Refere-se ao quão preciso e correto é o dado sobre a pesquisa em questão.
    - Vulnerabilidade: Pois a violação de dados importantes é uma grande violação.

- Volatilidade: Com as rápidas mudanças globais, quanto tempo resta para um dado se tornar irrelevante? Precisa-se de maneiras para avaliar o valor e disponibilidade dos mesmos.
- Visualização: Limitações como memória, escalabilidade, tempo de resposta são os principais desafios quando se tenta plotar esses dados, além da variedade de valores resultantes.

6. **Cite e explique os principais modelos de dados usados por bancos NoSQL.**
    - Documentos: Armazenam dados semi-estruturados e descrição dos dados no formato de documento.
    - Colunares: Organizam os dados da tabela em colunas ao invés das linhas.
    - Grafos: Organiza os dados como nós, que são como uma tabela de um database relacional, e arestas que representa a conexão entre os nós.
    - Chave-valor: Representa os dados em um simples formato de pares, com uma chave única associada a um valor.

7. **Explique o teorema CAP.**

    Existem muitas motivações para os bancos de dados NoSQL, como por exemplo um modelo mais adequado para os seus dados ou até facilitar alterações de schema, ou ir além disso e melhorar o desempenho e simplificar a replicação para ter a tão sonhada escalabilidade linear.

    Temos conhecimento que nenhum benefício vem sem custo, e o trade off arquitetural é descrito no conhecido teorema CAP.

    O Teorema CAP, assume e diz que em qualquer sistema distribuído *stateful* é preciso escolher entre **consistência forte [C]**, **alta disponibilidade [A]** e **tolerância a particionamento dos dados na rede[P]**.

    Segundo o teorema CAP, entre as três propriedades**, somente duas** podem ser garantidas ao mesmo tempo:

    Sistemas CP:

    Para sistemas que precisam da consistência forte e tolerância a particionamento (CP) é necessário abrir a mão da disponibilidade (*um pouco*).

    Sistemas AP:

    Por outro lado existem sistemas que jamais podem ficar offline, portanto não desejam sacrificar a disponibilidade. Para ter alta disponibilidade mesmo com um tolerância a particionamento (PA) é preciso prejudicar a consistência (eventual-consistency).

    Sistemas CA:

Os sistemas com consistência forte e alta disponibilidade (CA) (alta disponibilidade de um nó apenas) não sabem lidar com a possível falha de uma partição. Caso ocorra, sistema inteiro pode ficar indisponível até o membro do cluster voltar.

8. **O teorema CAP tem sido questionado. Ele ainda é válido? Por que? Explique os contra-argumentos ao teorema CAP.**

Enquanto o teorema é impecável na sua correção, a formulação pode ser enganosa sobre as implicações:
1. O teorema apresenta as três propriedades como iguais. Mas enquanto a consistência e a disponibilidade podem ser medidas em um espectro, a tolerância de partição é bastante binária. Pode-se variar a definição de Tolerância de Partição, mas, no final, só pode dizer que o sistema é compatível, possuindo ou não.

2. Se variarmos a definição de Tolerância de Partição, ela começa a se mesclar com a Disponibilidade. Uma Tolerância de partição temporária também pode ser chamada de indisponibilidade temporária.

3. A tolerância de partição só pode ser perdida em um ambiente hipotético onde nenhuma partição pode acontecer. Mas qualquer sistema real que perdesse a tolerância de partição não funcionaria corretamente e, portanto, a opção Disponibilidade e Consistência não deve ser considerada.

O teorema teria tornado suas implicações mais claras, se mencionasse Tolerância como uma propriedade dada, e dizer que, sob estas condições, apenas Consistência ou Disponibilidade podem ser garantido. Melhor ainda deve refletir o espectro de possibilidades, ou seja, deve falar de uma Consistência-Disponibilidade Desempenho em vez de escolha entre os dois.

Em um artigo posterior (Brewer, CAP, doze anos depois: como as "regras" mudaram, 2012), Brewer sugeriu outra melhoria que retrata o fator de tolerância de partição mais claro: A compensação entre Consistência e Disponibilidade só deve ser considerada quando a rede é particionada. Em qualquer momento em que a rede não está particionada, podemos ter ambos Consistência e Disponibilidade. Pode-se interpretar esse fato de que o sistema deve perder a tolerância de partição, desde que não há partição, e assim que ocorre uma partição de rede, ela precisa mudar sua estratégia e escolha uma compensação entre Consistência e Disponibilidade

9. **Quais os modelos de programação mais usados com Big Data? Explique brevemente cada um deles.**
    - MapReduce (Hadoop): Os dados são por padrão processados paralelamente entre os computadores de um cluster. Onde os programas Map analisam e processam subconjuntos dos dados e retorna resultados através de um fluxo como pares chave-valor ao passo que o Reduce os ordena e combinas os resultados do fluxo até o resultado final.
    - Functional (Spark): Programação funcional é um paradigma centrado nos dados e programação de interfaces baseadas em modelos de dados embutidos, resilientes e distribuídos. Spark foi desenvolvido sobre as limitações do paradigma MapReduce, que força programas distribuídos a escrever linear e coercivamente um fluxo de dados como uma cadeia conectada a tarefas de map e reduce.
    - Actor (Akka): É um modelo de programação distribuída e concorrente com tipos seguros baseados em Erlang.
    - Sql-Based (HiveQL): São query engines construídas em um sistema Hadoop e provisiona SQL como uma interface que ler entradas de dados no esquema definido e de forma transparente converte as queries em tarefas MapReduce conectadas como um grafo acíclico direcionado (DAG).
    - Estatístico e analítico(R): Combina a linguagem de programação S e com escopo léxico inspirado por schemas. E recentemente introduzido para processamento de contexto do Big Data para facilitar o desenvolvimento de aplicações estatísticas e analíticas.
    - Dataflow (Oozie): O modelo é programado como um grafo direcionado com operações e dependências como nós e arestas. Tem como ênfase o movimento dos dados e considera sua programação como uma série de conexões.
    - BSP (Google Pregel): Bulk Synchronous Parallel, é um modelo computacional e programação para desenhar algoritmos paralelos. São considerados como uma série de global super passos.
    - DSL de alto nível (Pig Latin): Vários frameworks que provem suas próprias Domain Specific Languages para escrever dados intensivamente em paralelo a aplicações para prover melhor o melhor modelo em certas condições.