# Use case for DynamoDB:

You understand the business problem and can predict what questions your data will answer

# You must understand access patterns:

Size – How much data is being written/read?

Shape – Data is stored/organized according to how it will be queried.

Velocity – what are peak query loads?

# Organization:

Related data stays together (one table)

Use sort key

Use carefully select partition key to distribute data among partitions

Global secondary indexes – enables multiple queries on data <- prefer this over local

My Use Cases:

* A customer wants to look at all of their previous orders
* A customer wants to look at a specific order (details and line items)
* A warehouse employee needs to see unfulfilled orders by precedence (oldest orders first)

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html>:

The partition key of an item is also known as its hash attribute. The term hash attribute derives from the use of an internal hash function in DynamoDB that evenly distributes data items across partitions, based on their partition key values.

The sort key of an item is also known as its range attribute. The term range attribute derives from the way DynamoDB stores items with the same partition key physically close together, in sorted order by the sort key value.

* A well designed application should only have one table (this was explicitly stated several times)
* Relational data can be modeled using the Adjacency List Design Pattern
* Write sharding can help distribute data across partitions (<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-partition-key-sharding.html>)