Scaling Out

Partitioning

Sharding splits data across multiple data stores in such a way that we can work out which information is on which host. For an in-depth discussion of partitioning see

Azure Best Practices For Data Partitioning

Table 1 Benefits of Partitioning

Scalability	Dividing data across multiple data stores prevents us being limited by the physical limits of a single store
Performance	Splitting data cross multiple data stores can lead to better performance as we need to search through a smaller amount of data on each partitioned store.
Flexibility	We can allocate different types of data to different types of data store. In this way the data store used is the one most appropriate for the type of data.
Availability	

There are several different ways of partitioning the data.

Horizontal Partitioning / Sharding

Each partition is separate data store. All data stores have the same schema and holds a subset of the data. The following sections describe some strategies for allocating subsets to partitions

Range Strategy

Vertical

Each partition holds a subset of the fields. Fields are divided according to how they are used e.g. frequently accessed fields might go into one partition.

Functional

The partitions are determined by the bounded contexts of the architectural solution. We might put orders in one partition and product definitions in another partition.

Key/Hash Based

Given N servers put the data on mod(key,n). As we add servers we need to repartition all the data which is expensive.

Directory based

Use a lookup table to prevent repartitioning as we add servers. The drawback of this approach is that the lookup can become a single source of failure and the extra level of performance can impact performance.