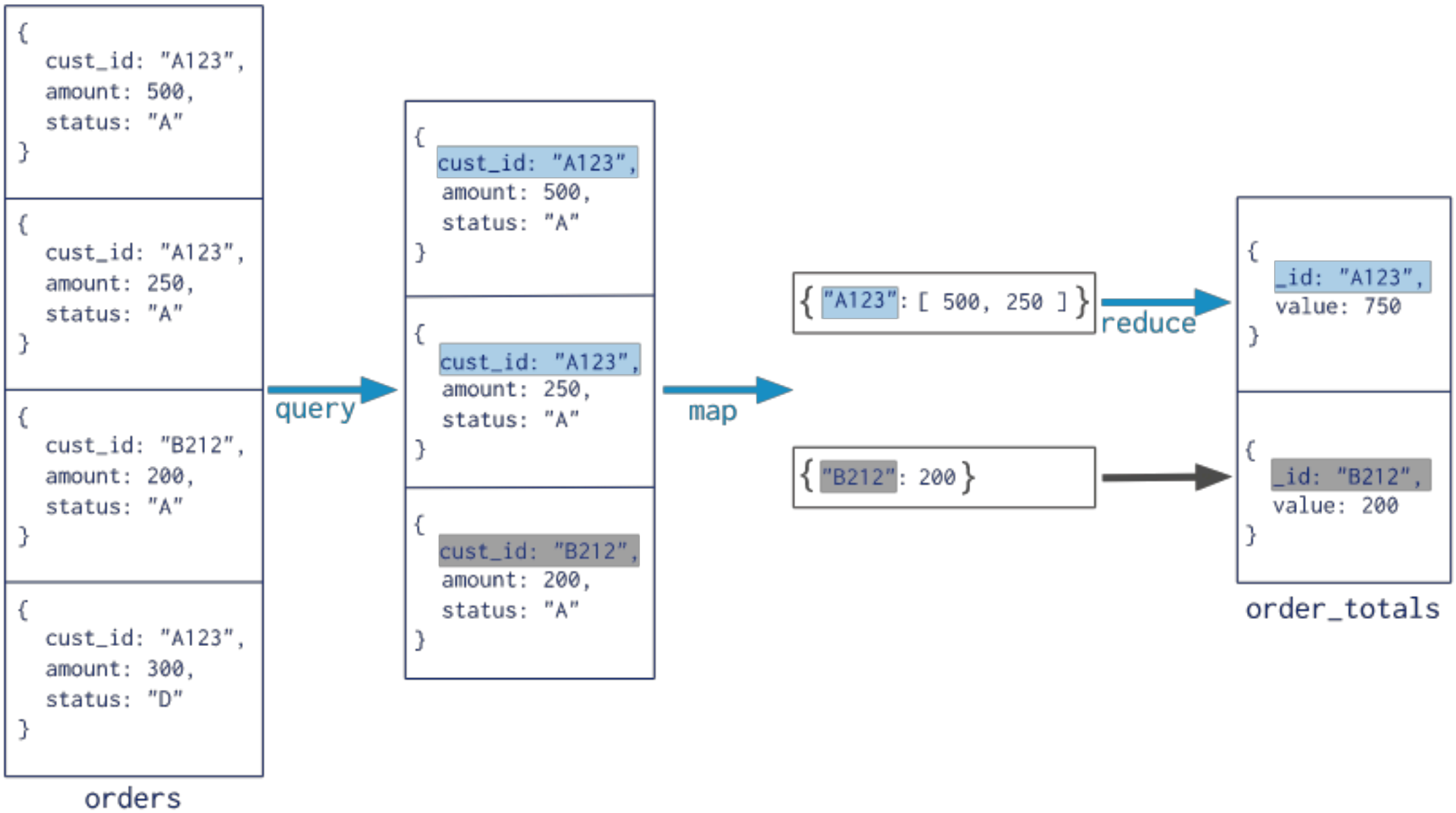
Collection

```
db.orders.mapReduce(
          map      ──►  function() { emit( this.cust_id, this.amount ); },
          reduce   ──►  function(key, values) { return Array.sum( values ) },
                        {
          query    ──►    query: { status: "A" },
          output   ──►    out: "order_totals"
                        }
                      )
```



This map-reduce operation applies the **map** phase to documents in the collection that match some query condition (i.e,. customer IDs).

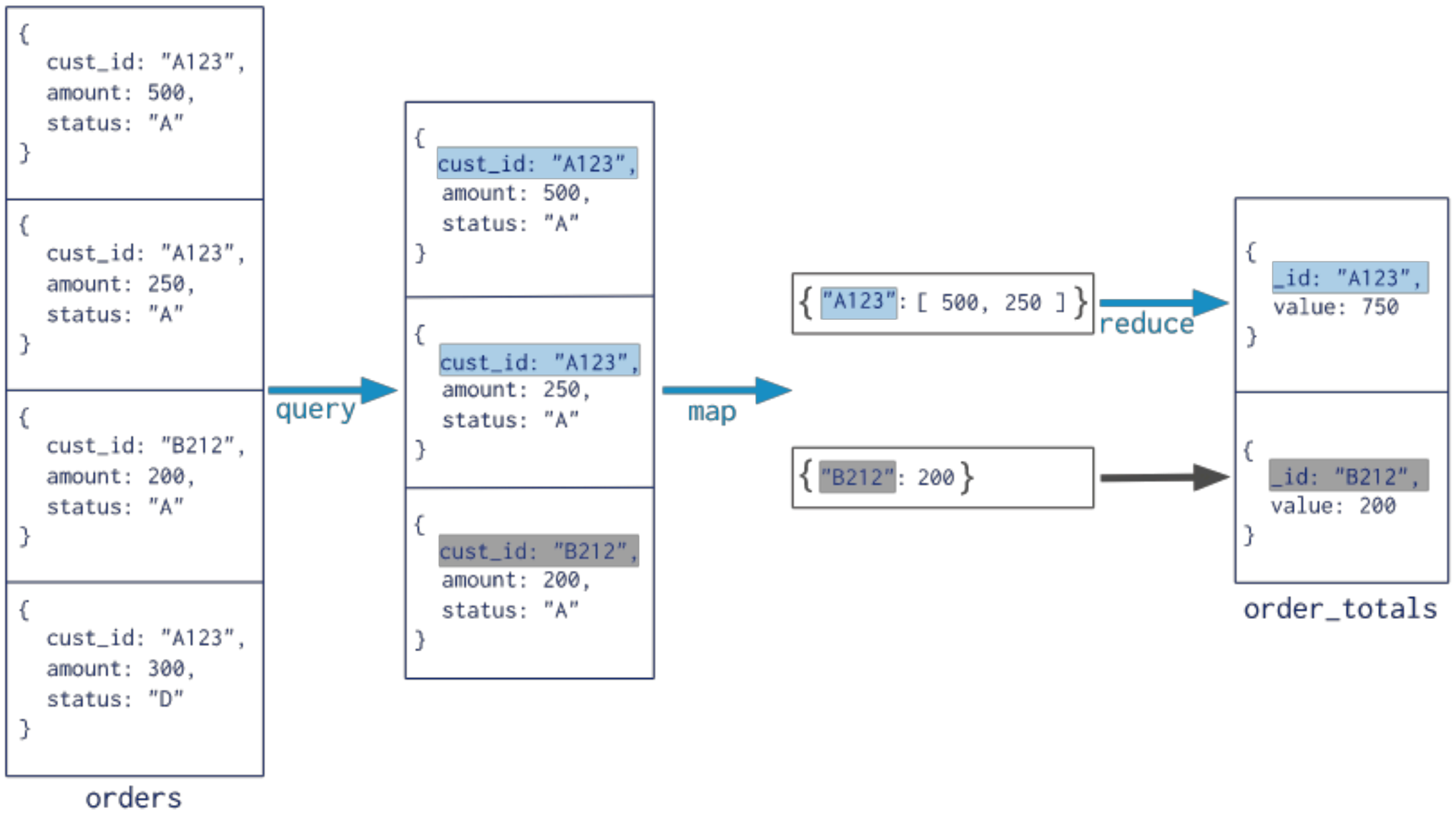
For those keys (customer IDs) with multiple values (i.e., orders), the **reduce** phase collects and condenses the aggregated data.

Finally, the results are aggregated and stored in some collection.

This map-reduce operation applies the **map** phase to documents in the collection that match some query condition (i.e,. customer IDs).

For those keys (customer IDs) with multiple values (i.e., orders), the **reduce** phase collects and condenses the aggregated data.

Finally, the results are aggregated and stored in some collection.



BY THE END OF THIS SESSION YOU SHOULD BE ABLE TO:

- Conceptual:
  - Identify the 4Vs of Big Data, defining each criterion.
  - Understand the difference between structured and unstructured data.
  - Differentiate between serial and parallel processing.
  - Explain parallelization in terms of its advantages in dealing with Big Data.
  - Identify the map and reduce phases in MapReduce and explain what is happening in both parts.

- Identify the map and reduce phases in MapReduce and explain what is happening in both parts.
- Explain parallelization in terms of its advantages in dealing with Big Data.
- Differentiate between serial and parallel processing.