SSIS optimization & large dataflows

ETL & ELT using SSIS

Three types of Data Flow Transformations

- Non-blocking (also: *synchronous*)
- Semi-blocking
  - Asynchronous
- Blocking
  - Asynchronous

Non-blocking transformation

- Start processing each row as soon as it arrives
- Is *synchronous*:
  - One row is emitted for every row received
  - Reuses input buffer
  - Faster than other components
- For example:
  - Derived column
  - Lookup
  - Data conversion
  - Conditional Split
  - Balanced Data Distributor
  - Multicast
  - Etc.

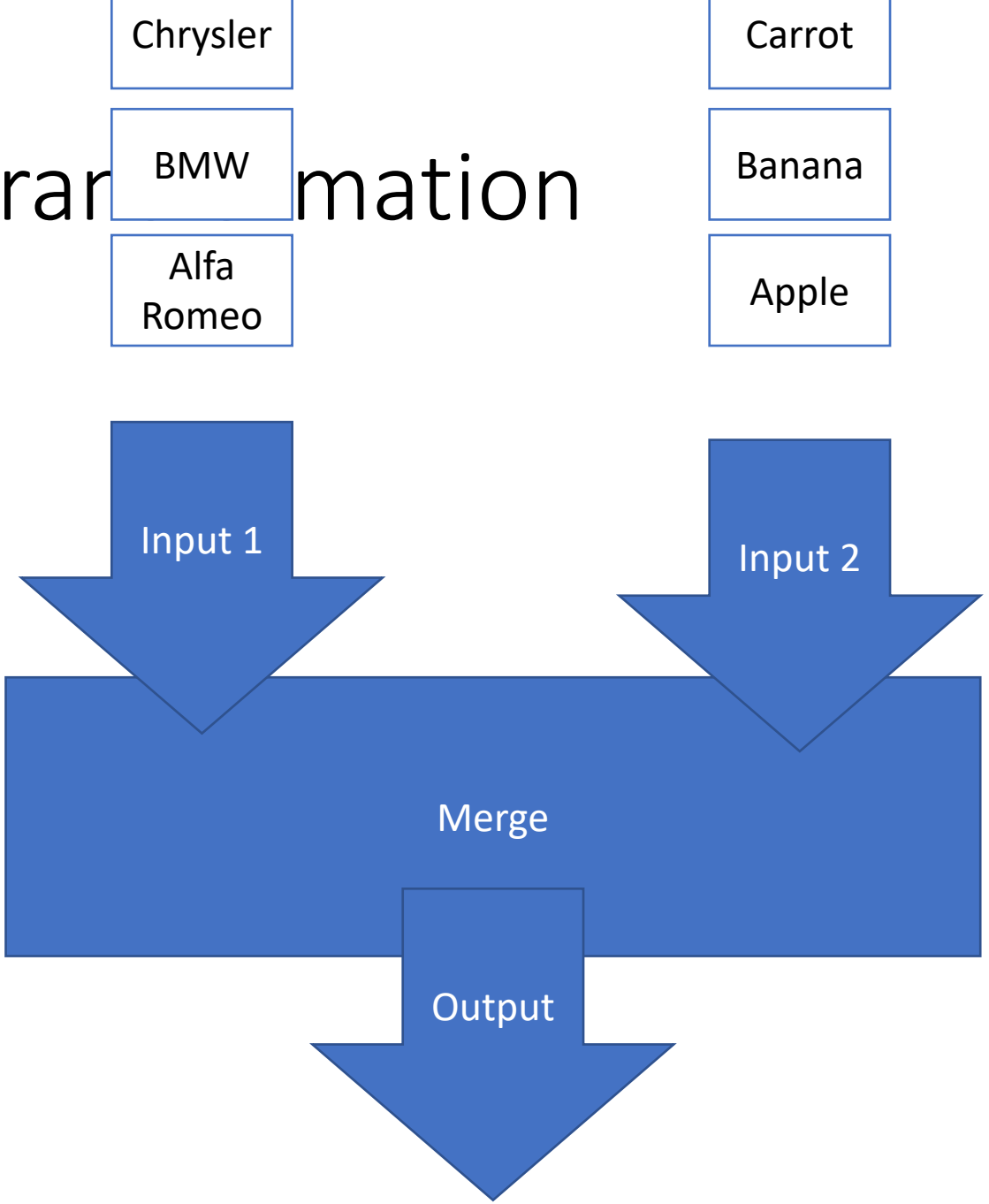
Blocking transformation

- Needs all input rows before it can emit rows
- For example:
  - Sort
  - Aggregate
- In case of sort:
  - Needed for some components
  - Better sort it at the database
    - … but be sure to use the right collation ☺ (is Née after or before Nee?)
  - Mark your inputs as “sorted” in advanced settings

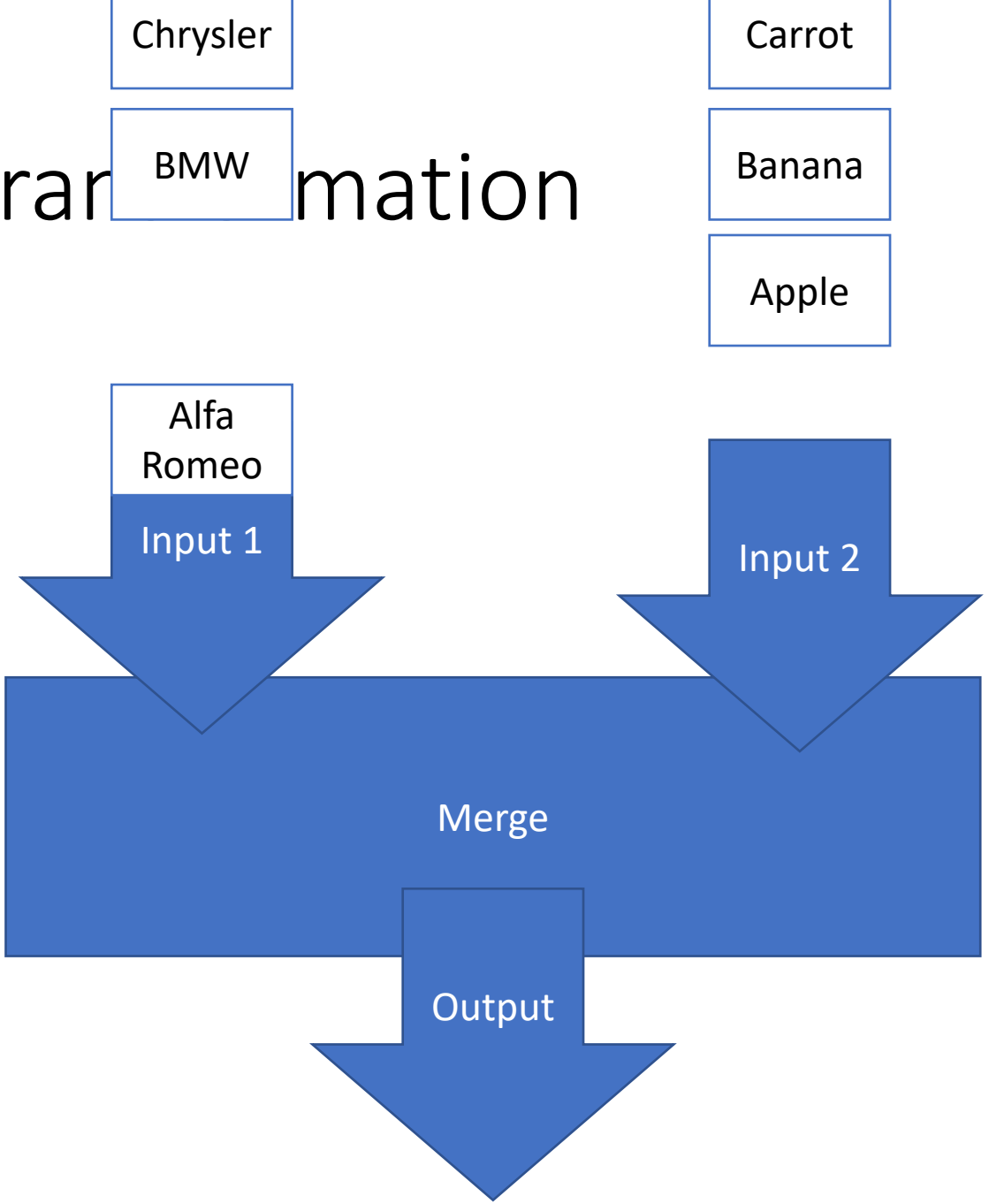
Semi-blocking components

- Need some, but not all input rows
- For example, the merge component:
  - Merges two sorted datasets
  - Final dataset is sorted as well

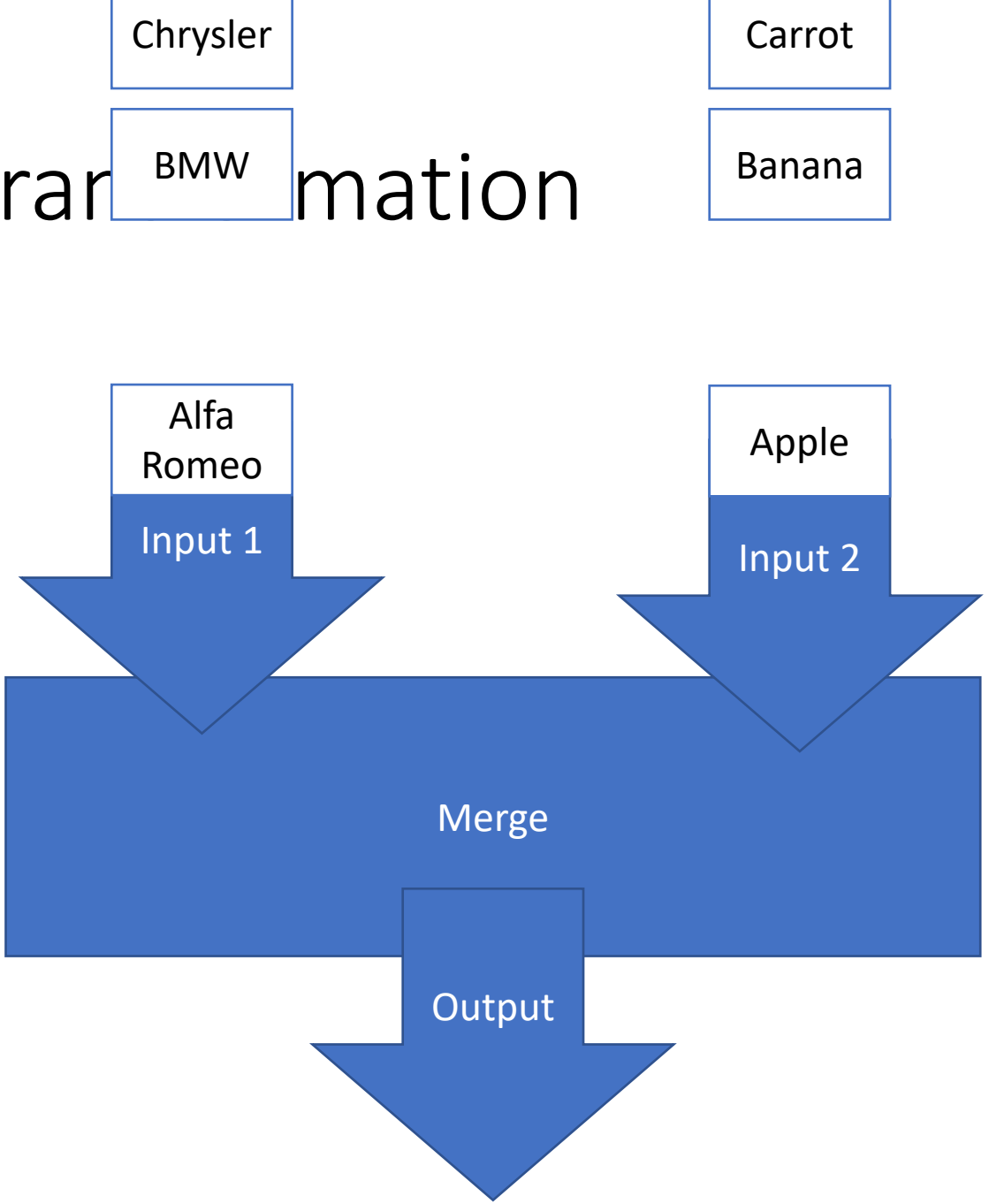
Merge transformation



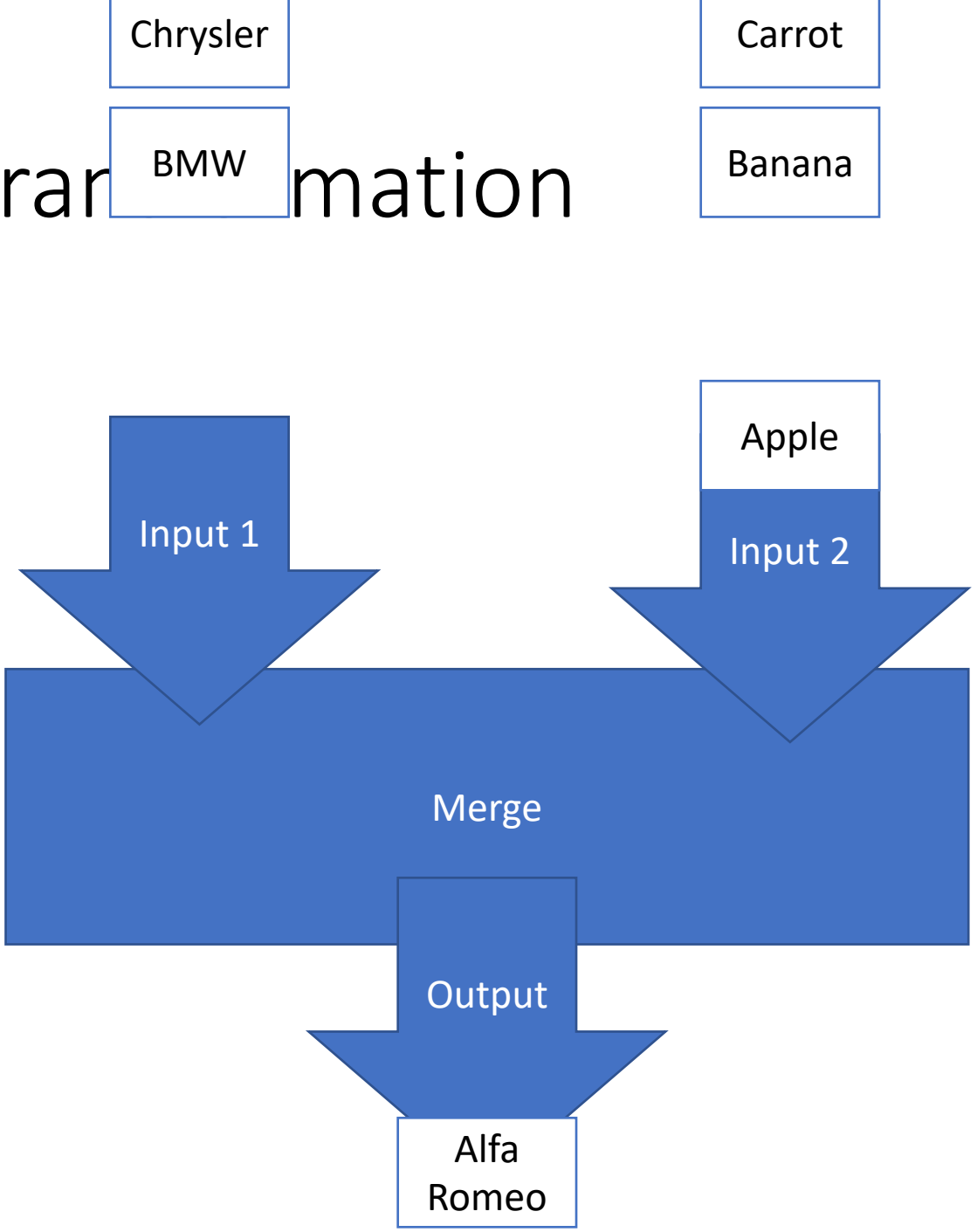
Merge transformation



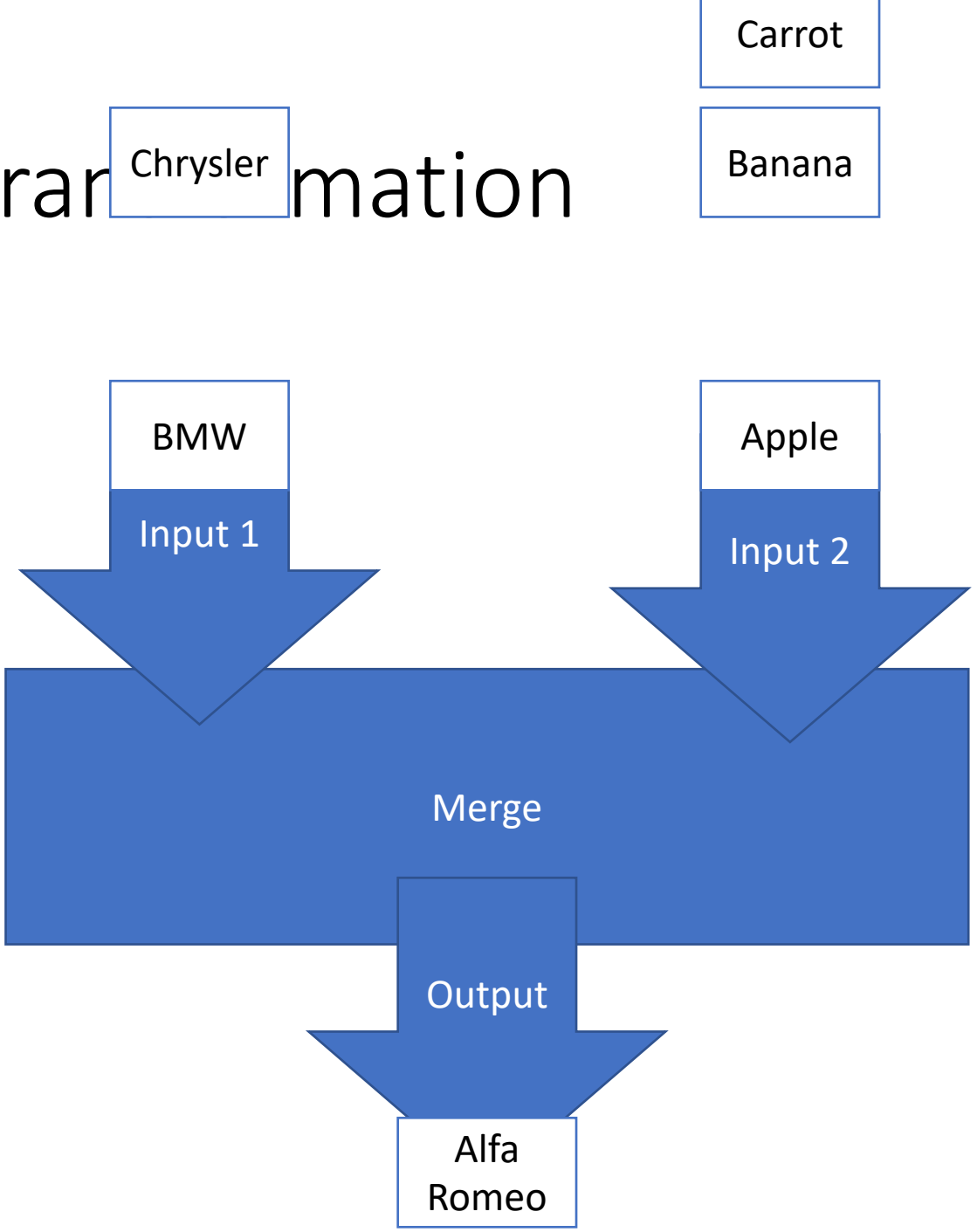
Merge tran mation



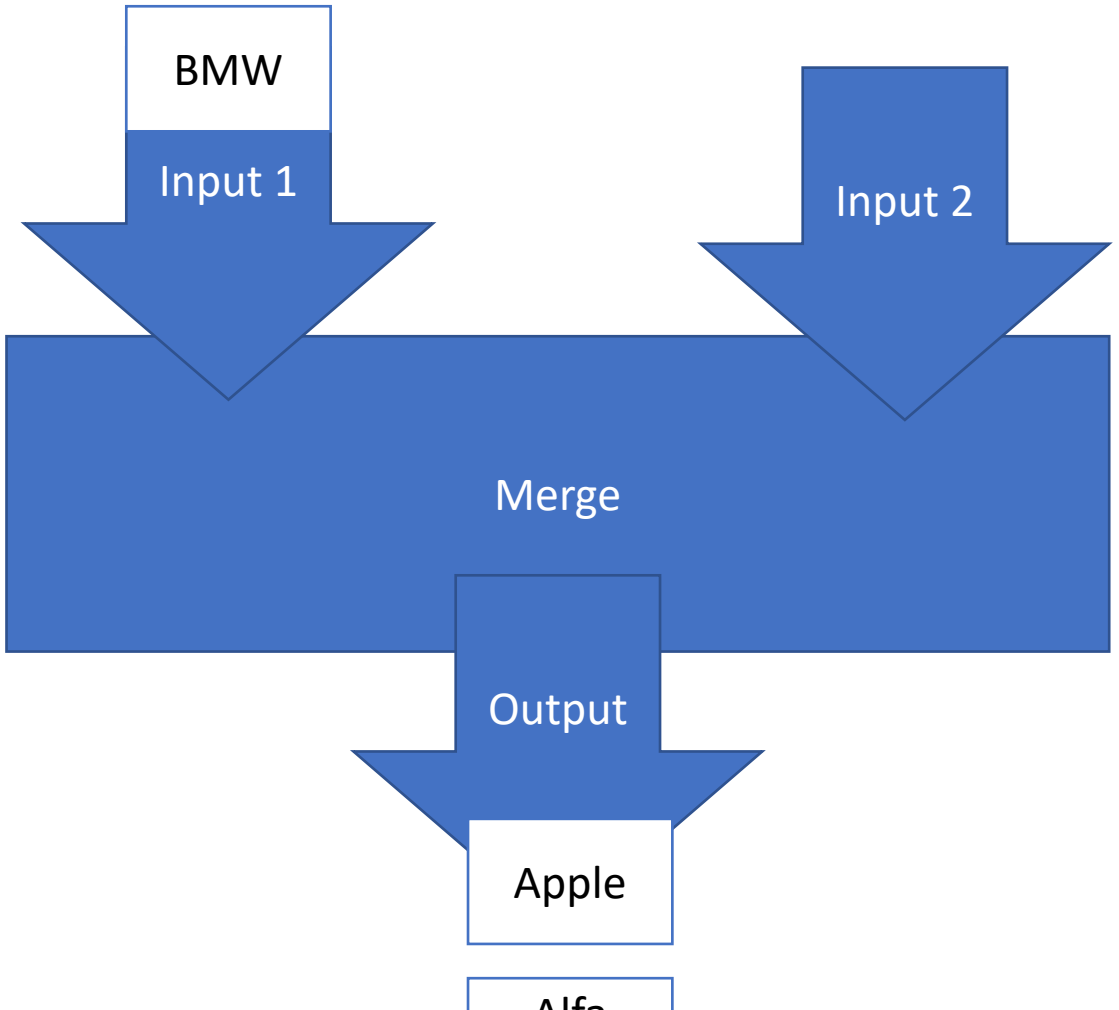
Merge transformation



Merge transformation



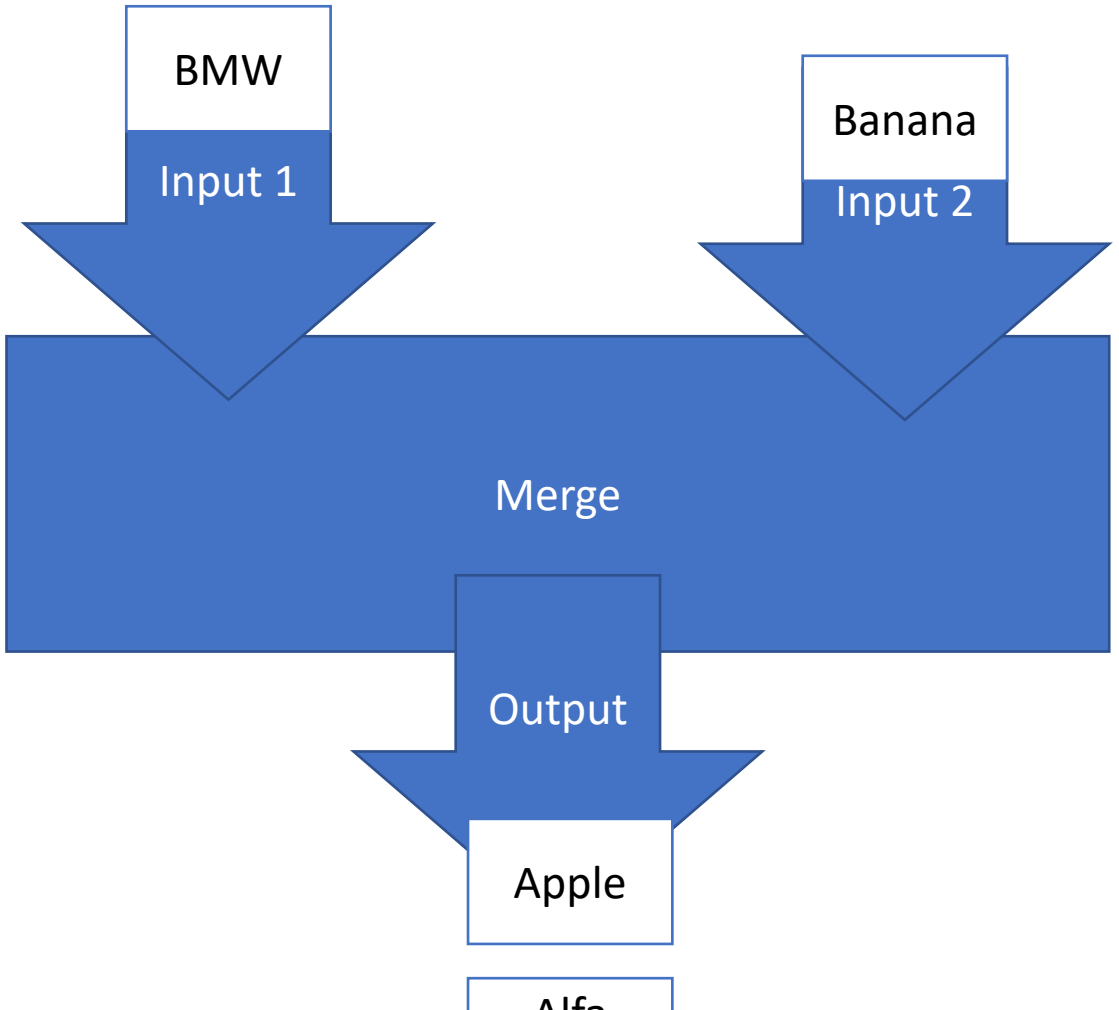
Merge transformation



Merge transformation

Chrysler

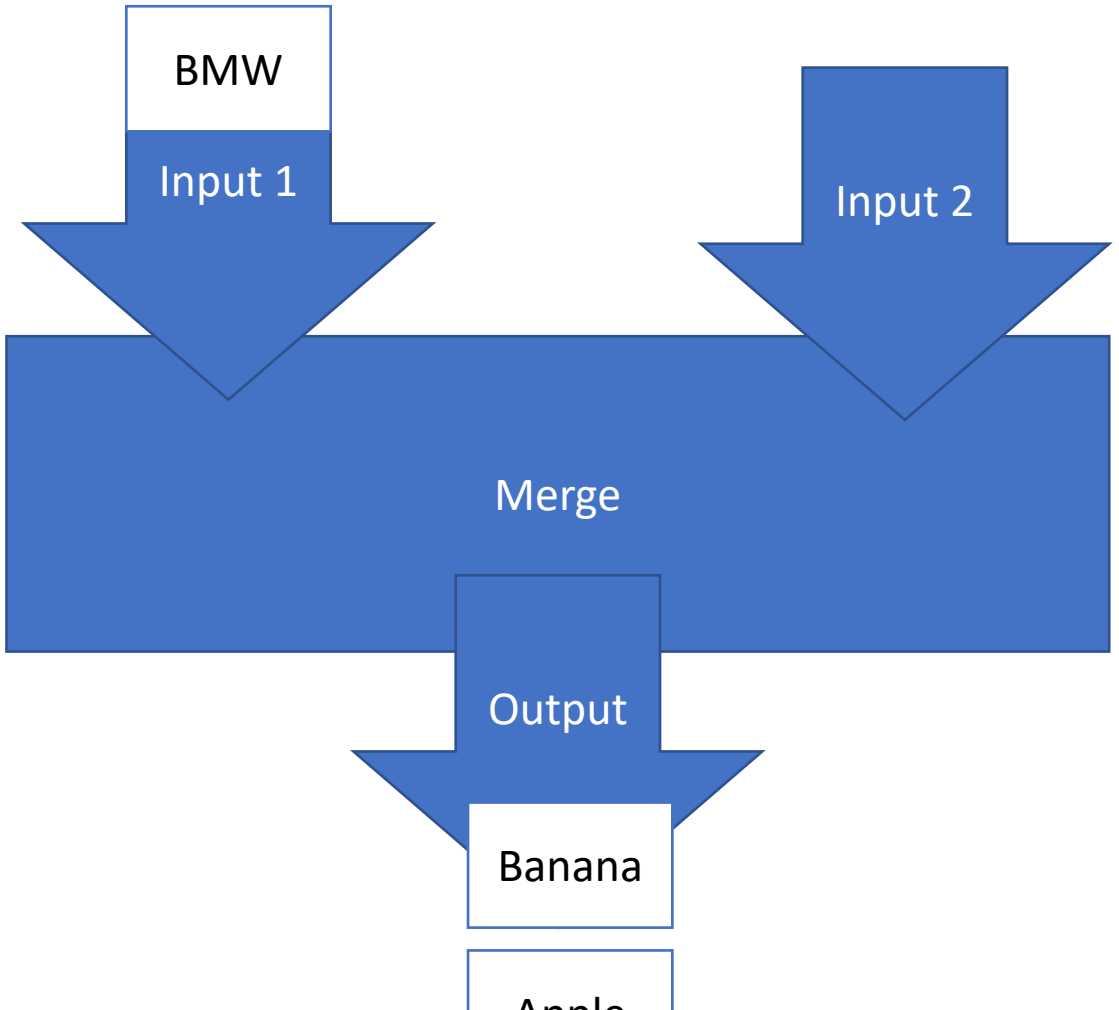
Carrot



Merge tran mation

Chrysler

Carrot



Semi-blocking components

- Need some, but not all input rows
- For example, the merge component:
  - Merges two sorted datasets
  - Final dataset is sorted as well
- Merge Join
  - Just like INNER / LEFT / RIGHT JOIN, but now in SSIS
- Union All

General advice for larger data flows

- If you need sorted datasets, sort them at the database
  - Be aware of the collation!
- Try to avoid semi-blocking and blocking components
  - They obstruct the “data pump”, and use way more buffer size
- Try to split up your data flows if they’re too large or complex
  - Just like constructing complex SQL transformations: would you benefit by persisting halfway?