

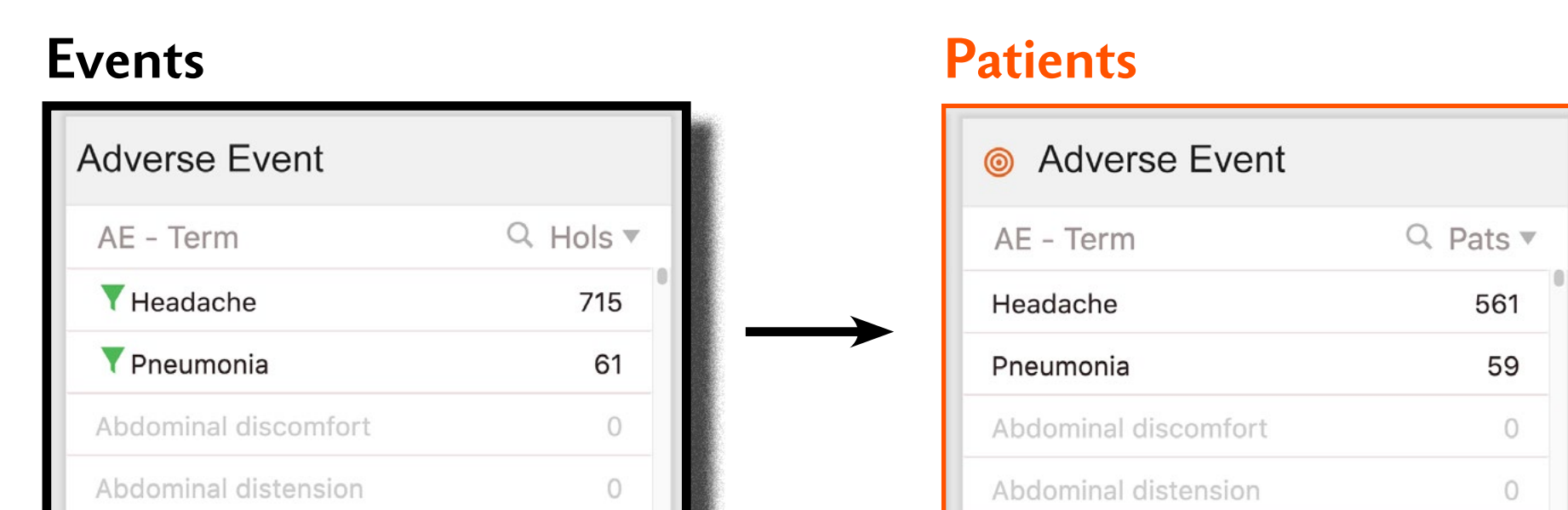
Introduction

Using a **graph database** for data storage and retrieval mimics the real world and how the human brain works. Thanks to the implementation of a new logic when querying a graph database, logical operators such as OR, AND and AND NOT can be utilized in a more efficient way.

Reflective Logic

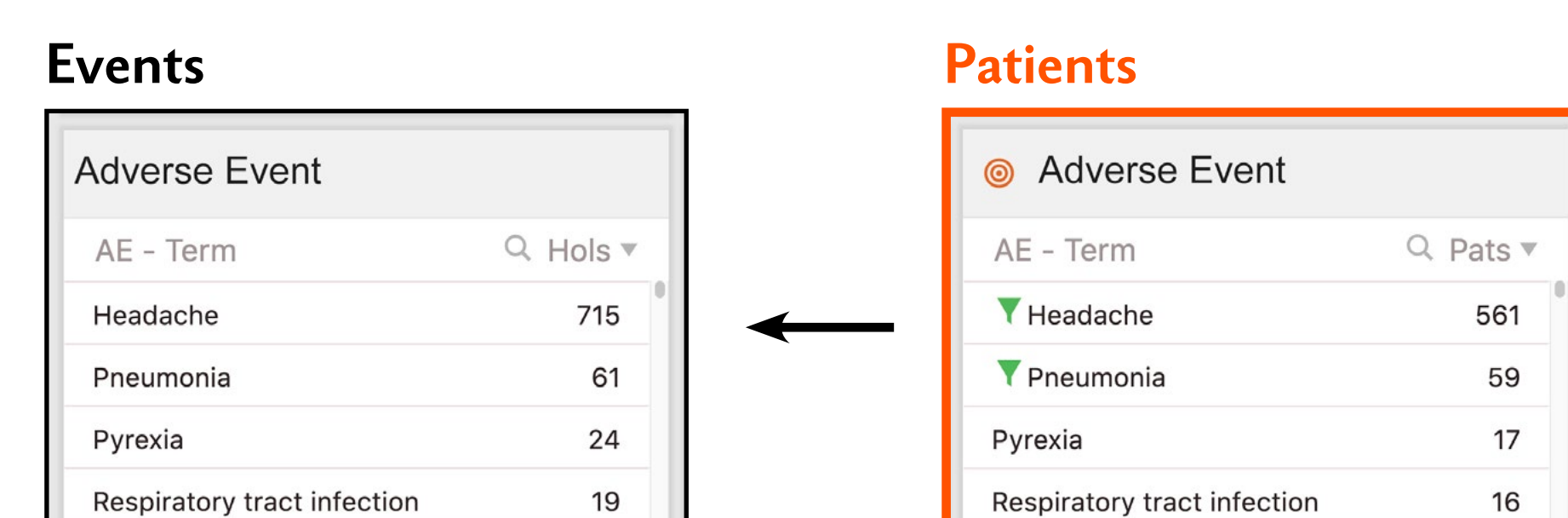
The **use of** Reflective Logic makes it possible for end users to look at data from different perspectives e.g., the patients' point of view. Depending on the context where the action is taken, different results are achieved i.e., the database queries are different.

Query 1 How many patients...?



Filtering on AEs in the list counting **events**, results in the count of patients that have experienced the selected AEs.

Query 2 What other events...?



Filtering on AEs in the list counting **patients**, results in all the AEs that the filtered patients ever have experienced.

Compared to Relational Databases

In **table-based tools** there is an implicit OR when querying multiple values within a field and an implicit AND between different fields.

The functionality when filtering on events in Capish Reflect is identical to the row-selecting slice-and-dice properties of traditional BI-tools. However, using the reflective logic together with logical operators creates new opportunities.

Reflective Logic makes database querying even more powerful.

The **Reflective Logic** opens up for using OR, AND and AND NOT when combining values in queries, both within and between fields. Using AND and AND NOT is dependent on defining 'what' should have both one AND the other, in this case patient.

 We call it Reflective Logic because you get two answers. The one you asked for and the one you will be looking for when reflecting over the first.

A. Filtering multiple values within the same field.

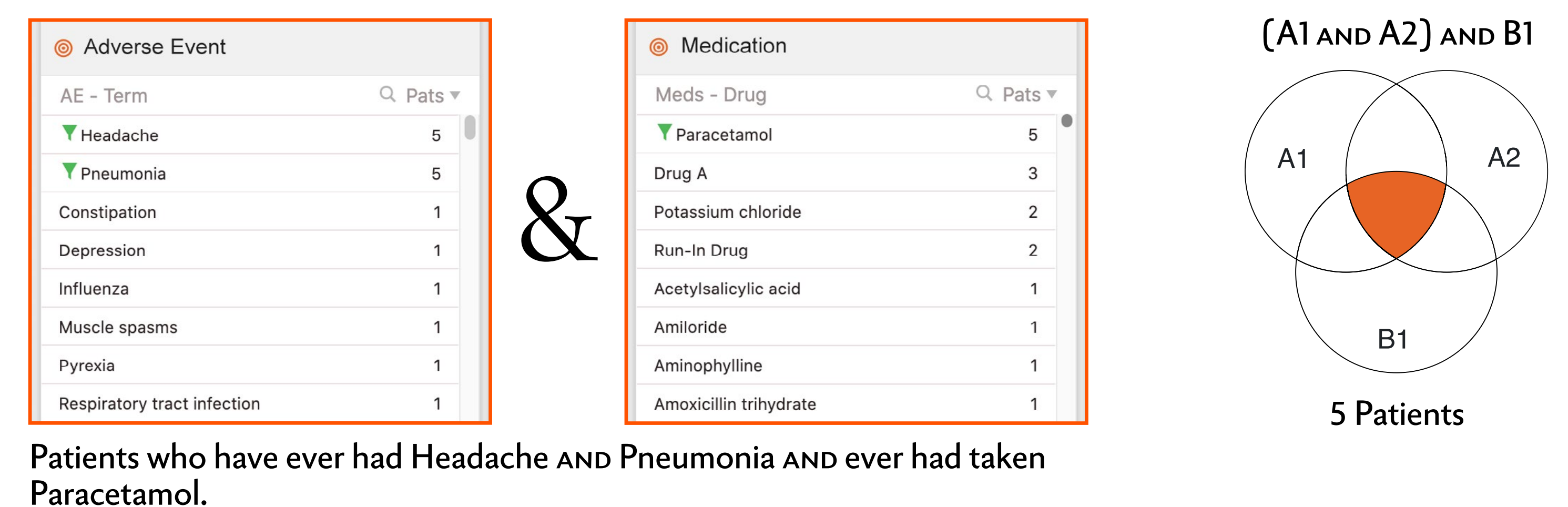


B. Filtering multiple values in different fields.

Different results are achieved dependent on how the query is created. Patients fulfilling the criteria are identified in both examples.

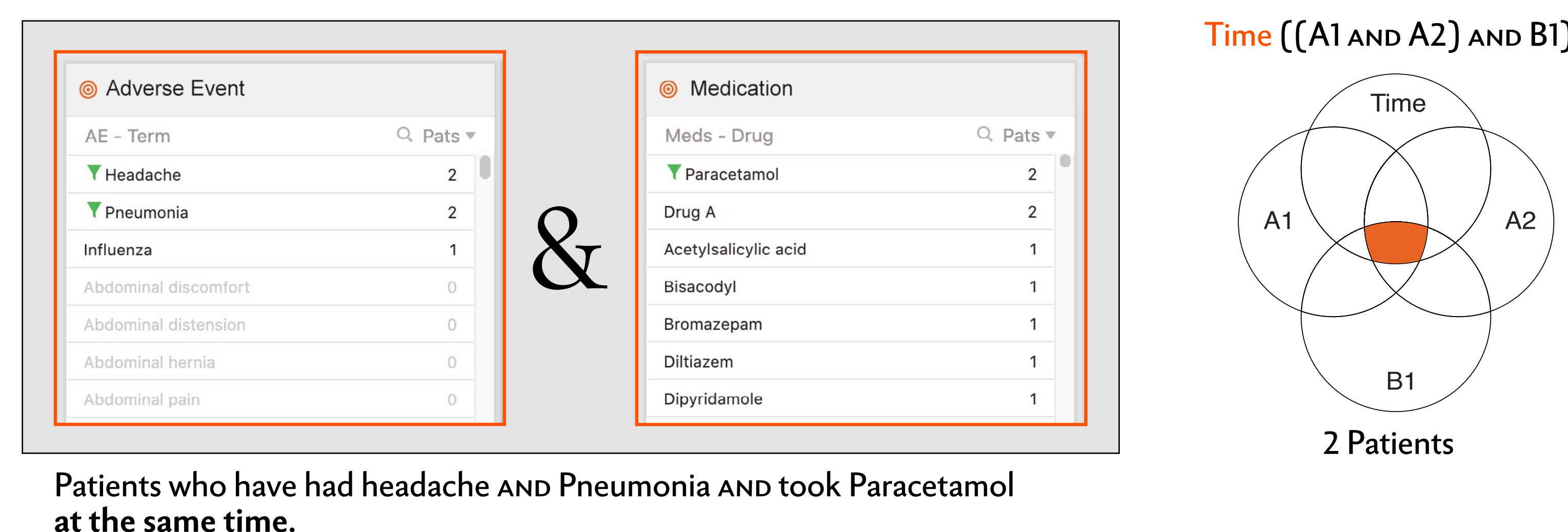
1. Filtering two different fields, *independent* of time.

The query looks for events that ever have happened but not necessarily at the same time.



2. Filtering two different fields, *dependent* of time.

In this case the query looks for events that have happened at the same time.



In the illustrations on the left, one conclusion is that only one of the 21 patients has ever had a Respiratory tract infection as well as Headache and Pneumonia. Since the query is not time dependant the events must not have happened at the same time.

From these four queries of data in a field, cohorts can be created in an easy way for further explorations as well as looking at data for an individual patient.

By combining multiple values in *different* fields the creation of even more complex queries is made possible.

Conclusion

The **value of this** approach is that it enables searching and getting answers to very complex questions in an easy way.

Being able to look at data from e.g., the patients' perspective gives the user a lot of information with little effort. The flexibility makes it possible to gather patients in different cohorts to explore the patients more deeply, individually or as a group.

To narrow down the patient material by using logical operators in this way can be very time-saving as the users are free to create the cohorts on the fly. It is ideal for exploring causality and find connections.

...with Capish you can.