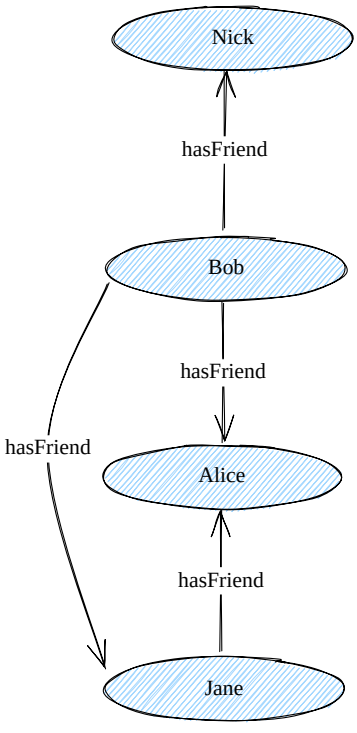
Detailed Example of Knowledge Graph

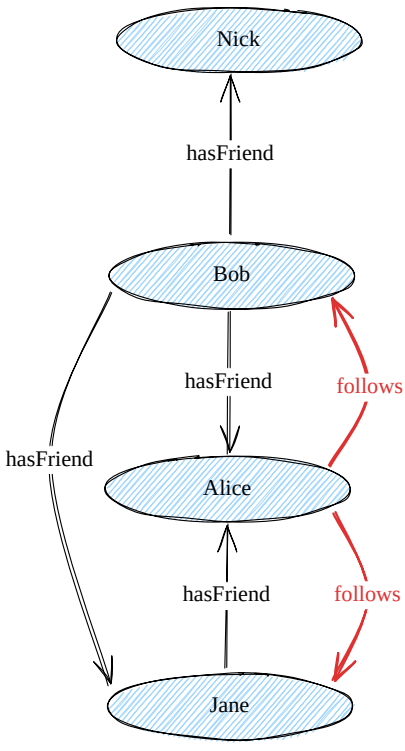
Based on the Social Media example used within the blog, here we demonstrate the difference in flexibility to update a graph data representation of your digital twin with respect to a relational table representation

Graph representation of a Digital Twin

Flexibility in Structure: Graphs are inherently flexible and can easily accommodate new types of nodes (entities) and edges (relationships) without requiring a predefined schema. This means you can continuously add new data and relationships as they emerge, without needing to restructure the entire graph.

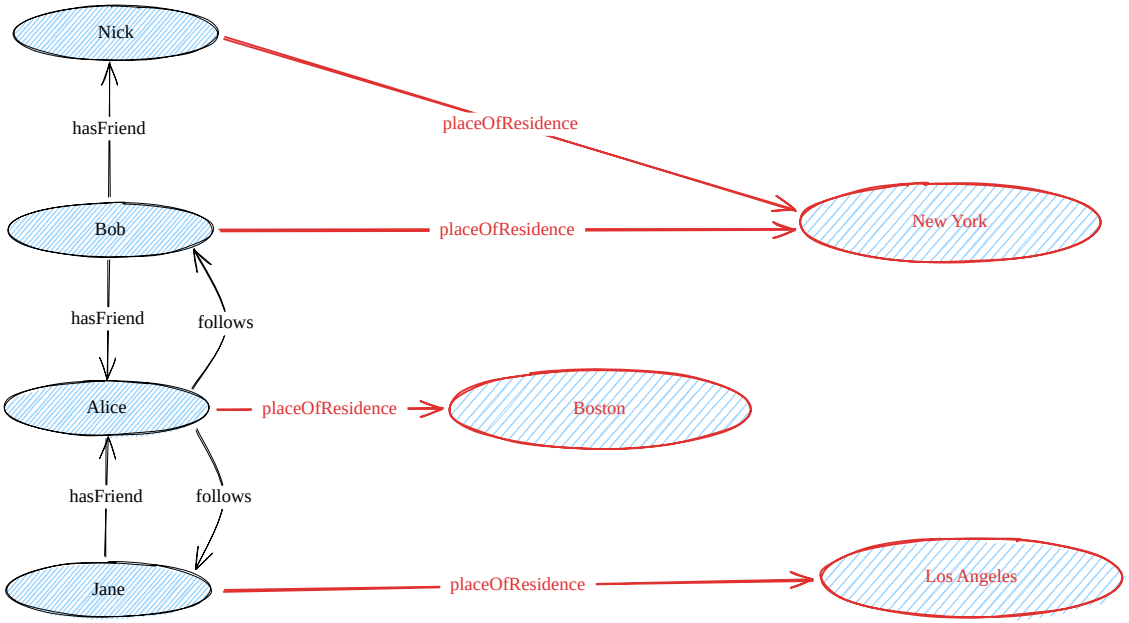
Example: Imagine a social network graph where nodes represent people and edges represent relationships (e.g., friends). If you want to add a new type of relationship, like "follows" (similar to Twitter), you can simply add this new edge type without altering the existing structure.

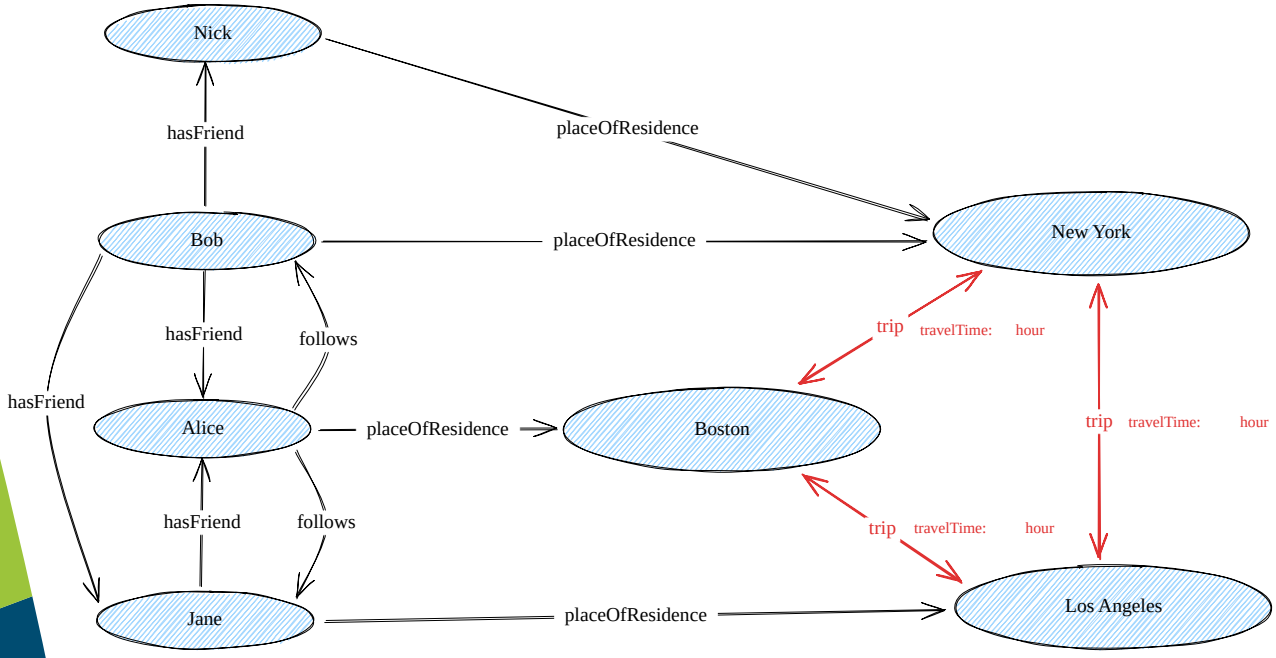




Dynamic and Evolving: Graphs can evolve over time as new data is added. This makes them ideal for applications where the data model is expected to change or grow, such as digital twins, where new sensors or data sources might be integrated over time.

Example: Now assume you also want information on the place of residence of each person. In a graph you would add nodes representing the cities. And to indicate a person has its residence in a city, you add a relationship between the person's node and the cities node with the name "placeOfResidence". Suppose that much later, as a requirement of another use-case you need information on the travel time between cities mutually. A simple way to do that would be to add a "trip" relationship between two cities with a property "travelTime" that stores the travel time in hours for example. Although the trip relationship was used in another use-case, the addition of this information also allows to determine how much time it needs for two friends to travel to each other. For all this, no adaptation was required of existing data and hence existing applications. Simply new data/information was added enabling new, and sometimes unforeseen, functionality to your Digital Twin.





Royal HaskoningDHV
Enhancing Society Together

Relational Table representation of a Digital Twin

Fixed Schema: Relational tables require a predefined schema, meaning the structure of the data (tables, columns, relationships) must be defined upfront. Adding new types of data or relationships often requires altering the schema, which can be complex and time-consuming.

> *Example: In a relational database for a social network, you might have a table for persons and another for friendships. The later contains a mapping from one person (id) to another. If you want to add a new type of relationship, like "follows," you would need to create a new table or modify the existing friendship schema, which can be disruptive.*

Limited Relationship Representation: Relational tables are not as effective at representing complex, interconnected relationships. While you can use foreign keys to link tables, querying and navigating these relationships can become cumbersome, especially as the number of relationships grows.

> *Example: Identical to the previous use-case, you want to extend the data with information on the place of residence of a person. A very common way to do that is to augment the already existing persons table with an extra column with name "placeOfResidence" where the cities name is stored. This works fine and to find the place of residence of a person is a simple and very fast evaluation. However, to add (at a later stage) the travel time between cities is quite cumbersome. Cities do not have their own table in this case, because you could not foresee the fact that they were going to be instances of their own, requiring characteristics and mutual relationships. In other words, a new cities table needs to be introduced, the people table needs to be adapted (the string now becomes a link to the cities table) and a new trip table needs to be made to represent the travel time. In time, the relational structure is slowly moving towards a graph-like structure. However, querying and maintaining this structure is much harder than this would be if a graph database was used.*

Want to know more about our expertise?

Contact Ben Lomax Thorpe, Digital Twin Leading Proffesional

ben.lomax.thorpe@rhdhv.com

royalhaskoningdhv.com

Royal HaskoningDHV
Enhancing Society Together