Towards Data Visualisation based on Conceptual Modelling

Peter McBrien[1] and Alexandra Poulovassilis[2]

[1] Dept. of Computing, Imperial College,
180 Queen's Gate, London SW7 2BZ, p.mcbrien@ic.ac.uk
[2] Birkbeck Knowledge Lab, Birkbeck, University of London,
Malet Street, London WC1E 7HX, ap@dcs.bbk.ac.uk

Abstract. Selecting data, transformations and visual encodings in current data visualisation tools is undertaken at a relatively low level of abstraction - namely, on tables of data - and ignores the conceptual model of the data. Domain experts, who are likely to be familiar with the conceptual model of their data, may find it hard to understand tabular data representations, and hence hard to select appropriate data transformations and visualisations to meet their exploration or question-answering needs. We propose an approach that addresses these problems by defining a set of visualisation schema patterns that each characterise a group of commonly-used data visualisations, and by using knowledge of the conceptual schema of the underlying data source to create mappings between it and the visualisation schema patterns. To our knowledge, this is the first work to propose a conceptual modelling approach to matching data and visualisations.

1 Introduction

Current data visualisation approaches base their visualisations on simple table data presentations, and fail to capture the full schema knowledge when the underlying data source is a structured database, such as a relational database. Furthermore, creating visualisations requires a fresh data mapping effort for each visualisation that is created, be it programmer effort or end-user effort. We propose an approach that addresses these problems by firstly defining **visualisation schema patterns** that characterise each distinct (from a data representation capability) group of commonly-used data visualisations, and secondly that uses the conceptual schema of the underlying data source to create mappings between the data schema and the visualisation schema patterns. The benefits of this approach are firstly that we use the full knowledge of the conceptual model of the underlying data to identify which are feasible visualisations for that data, by matching the data schema with the set of visualisation schemas; and secondly, once this mapping is in place, the creation of actual visual charts can utilise the mapping to extract data, drill-down, roll-up, pivot, switch visualisation *etc.* To our knowledge, ours is the first work to propose a conceptual modelling approach

to matching data and visualisations. We refer readers to [3] for a review of related work on visualisation tools, taxonomies, recommendation, and languages for manipulating graphical data.

2 Motivating Example

A fragment of the Mondial database [2] is illustrated in the ER diagram in Figure 1. It describes a schema about countries (including the current population), the history of a country's population in the weak entity country_population, and provinces in countries). For some countries, data about the GDP of the country is recorded in the subset entity economy, the attributes of which are all optional, indicated by the use of a question mark. Also recorded is which continent or continents a country belongs to: most countries will belong 100% to one continent; but the cardinality constraint of 1:2 allows some (e.g. Russia, Turkey) to spread over two continents, with the percent attribute of encompasses recording the proportion of their land area that belongs to each continent.

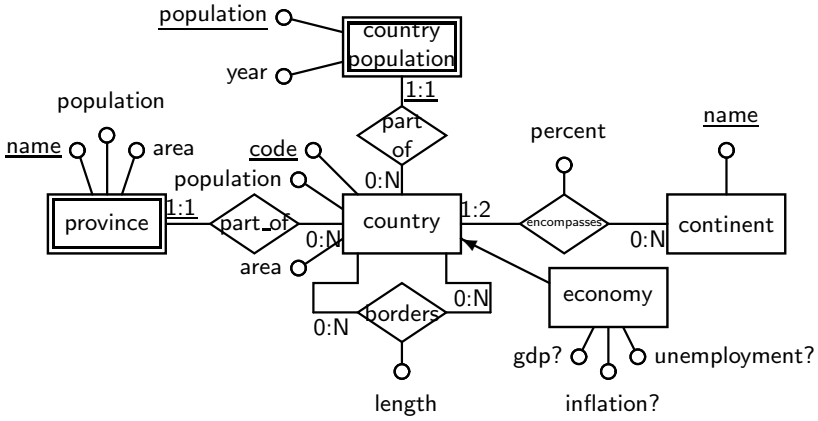
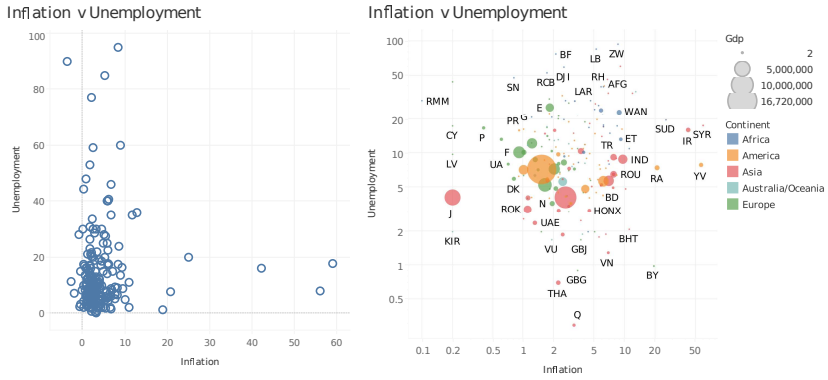


Fig. 1. ER schema of a fragment of the Mondial database

Suppose we wished to explore the relationship between inflation, unemployment, and GDP in countries. We could first extract a table of data with scheme (country, inflation, unemployment, gdp), where country corresponds to the key attribute code of the country entity in Figure 1, and without null values for inflation, unemployment, and gdp. Importing that table to Tableau, and choosing to represent countries as a 'dimension', and putting the inflation and unemployment figures on the x and y axis, produces the chart shown in Figure 2(a).

We see that, because of a few outlying data values, most of the data appears in a small cluster to the bottom left of the diagram and is largely illegible. No use has been made of the fact that the data distribution can easily be determined to be skewed, and hence an alternative scaling could have been used. Furthermore, no suggestion is made on how to include the gdp column of the table, despite the fact that this is numeric-valued, which would suggest displaying its data using



(a) Inflation v Unemployment (b) Addition of GDP and Continent

Fig. 2. Presentation of country data in Tableau

a graphical construct suitable for representing ranges of numbers. Figure 2(b) shows the result of a user (manually) determining that a logarithmic scale will better spread the data relating to the relationship between inflation and unemployment, and that the data in gdp can be used to scale the size of the circles, to make a bubble chart. Figure 2(b) also colour-codes countries by their continent — as suggested by the database schema, which connects countries to continent via a relationship with restricted (upper bound 2) cardinality.

3 Visualisation Schema Patterns

Our starting premise is that each instance of an entity in the database is associated with one or more graphic elements, which in visualisation are usually classified [1] as **marks** (points, lines, areas, *etc*) or **channels** (colour, length, shape, coordinate, texture, orientation, movement, *etc* of a mark). An attribute value of an entity, or the participation of an entity in a relationship, is associated with a dimension of the visualisation, and the process of visualisation is about choosing the correct graphic elements for a given schema.

Taking an approach similar to Tableau, we identify the following two major types of dimensions (which differ from the discrete and continuous classification found in [5]):

- **discrete dimensions** have a relatively small number of distinct values, that may nor may not have a natural ordering; they are used to choose a mark or to vary a channel of a mark.
- **scalar dimensions** have a relatively large number of distinct values with a natural numeric ordering (e.g. integers, floats, timestamps, dates); these are represented by a channel associated with a mark.

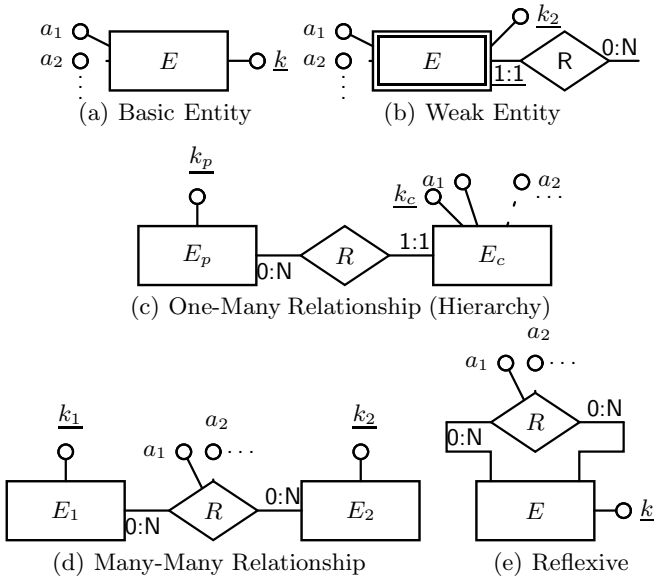
When a dimension is represented by a **colour channel**, then if it is a discrete dimension it lends itself to using a colour key, where each colour represents a

discrete value. Alternatively, if it is a scalar dimension, then a spectrum of colours can be used to represent a range of values. Hence, in our descriptions below, when we talk of a colour we assume the ability to automatically choose between these two representations based on the type of the dimension.

Scalar dimensions are **evenly distributed** if their values are (roughly) spread evenly over the entire range of values in the dimension (many visualisations struggle to represent data where most data is in a small range of values and there are some outlying values).

As is well known [1], what we are naming discrete or scalar dimensions may have specific real-world characteristics, and may for example be a **geographical**, **temporal**, or **lexical** dimension. This characterisation then may suggest specific visualisations for their representation (*e.g.* a map, time slider, word cloud, *etc*). However, in this paper we focus on what assistance can be given to the visualisation process by the knowledge represented in the schema of the data, and hence we only consider these real-world characteristics if required for the use of a particular visualisation. Indeed our work should be viewed as providing assistance to existing visualisation techniques, to be used where data is sourced from a structured database. Our work is therefore complementary to aspects such as task-based visualisation design and interaction during design.

In the following subsections we present successively more complex visualisation schema patterns, and the visualisations that they encompass. Our survey of visualisation techniques has so far not found any visualisations that require more complex schema patterns than those presented here, and in particular none that require a pair of relationships to be considered together.



(a) Basic Entity (b) Weak Entity

(c) One-Many Relationship (Hierarchy)

(d) Many-Many Relationship (e) Reflexive

Fig. 3. Visualisation Schema Patterns for Data Visualisations

3.1 Basic Entity Visualisations

An ER entity can be regarded as a conceptual modelling of a relational table. Many visualisations are designed to represent such tabular data, so we begin by identifying a category of visualisations that are suitable for representing an entity with its keys and attributes. This 'basic entity' visualisation schema pattern is illustrated in Figure 3(a), where it should be noted that the key attribute k might be inherited from a parent, such as economy in Figure 1 having an inherited key code from country. Many visualisations fit into this category, and we list below a sample to illustrate the way in which different features of each visualisation are represented in our approach (more are given in [3], e.g. choropleths, word clouds).

- Basic **bar charts** represent instances of an entity E (identified by the value of k) as bars, with the length of the bar determined by the value of an attribute a_1. Hence a_1 should be a scalar attribute.
- A **calendar chart** (found in both D3 and Google Charts) represents instances of E according to a date-valued attribute a_1.
- In **scatter diagrams** (such as in Figure 2(a)), each point represents an instance of E, and two dimensions a_1 and a_2 are used to plot its x and y coordinates. Optionally, a third dimension a_3 can be used to colour it.
- In **bubble charts** (such as in Figure 2(b)), each bubble denotes an instance of E; two dimensions a_1 and a_2 are used to plot its coordinates, and a third dimension a_3 its size. A fourth dimension a_4 may be used to colour it.

The table below summarises the above analysis, where $|k|$ denotes the number of distinct values of the key k. The upper cardinality of 100 shown in relation to the bar chart is subjective, and aesthetics-driven; it would be user-configurable in any implementation.

Basic Entity Visualisations			
Name	$\|k\|$	mandatory	optional
Bar Chart	1..100	a_1 scalar	-
Calendar	1..*	a_1 temporal scalar	-
Scatter Diagrams	1..*	a_1, a_2 scalar	-
Bubble Charts	1..*	a_1, a_2, a_3 scalar	a_4 colour

Note that all of the above visualisations (and indeed those listed in the following subsections) may have additional temporal scalars represented by time sliders, and discrete scalars represented by snapshot or paging options.

In our approach, visualisation schema patterns are used in conjunction with the database schema to guide the process of choosing a visualisation, by finding sub-graphs of the database schema that match each visualisation schema. Although this is an instance of the (NP-complete) subgraph isomorphism problem, the query graph (i.e. the visualisation schema) will be small and hence we anticipate fast execution times using state-of-the-art algorithms such as [4].

For example, starting with the schema in Figure 1 and matching Figure 3(a) against it, a match is found with the entity country, with k matching code and

choices area and population for the scalars a_1 and a_2. The user can therefore be offered a bar chart or scatter diagram as a visualisation of the data.

3.2 Weak Entity Visualisations

A particular form of compound key (often arising from the representation of weak entity data in an ER schema) identifies a family of visualisations where one part of the key, k_1 (the key of the entity that the weak entity is attached to) identifies a set of tuples, and the second part of the key, k_2, identifies a tuple in the set. The visualisation schema pattern for this is shown in Figure 3(b), where it should be noted that k_1 would match a key relationship in the data; for example, in Figure 1 if E matched province then k_1 would match part_of and hence be based on the code of country.

The values of k_2 must lie within a similar range of values for all instances of k_1 (so as to make their visualisation in one chart meaningful). Also, we say that the values of k_2 are **complete** with respect to k_1 if it is the case that the same set of values appears for k_2 for each value of k_1. For example, the weak entity country_population in Figure 1 meets the range requirement since the dates for population figures range over a period of less than 200 years, but it fails the completeness test since the years in which population figures are available vary from country to country. By contrast, the province entity fails the range test, since the names of provinces are almost entirely disjoint with those of countries.

As with the basic entity visualisation, there are many visualisations suited to present the weak entity visualisation, a selection of which are listed below, together with a summary table:

- In a **line chart** each line represents a distinct value of k_1; k_2 represents a scalar dimension to be plotted along the x-axis; and a_1 must be a scalar dimension to be plotted along the y-axis. XY variations allow an additional dimension a_2 to be added to the y-axis.
- In a **stacked bar chart**, distinct values of k_1 are represented by a bar, with one of the elements in the stack representing a value of k_2, and the length of the bar determined by a scalar dimension a_1. Each value of k_1 should appear with the same (or almost the same) set of values for k_2 (the completeness property) so that the elements in each stack can be compared.
- In a **spider chart**, each ring represents a value of k_1 and each spoke a value of k_2; the intersection of the ring with a spoke is determined by a_1.

Weak Entity Visualisations					
Name	$\|k_1\|$	$\|k_2\|$	complete	mandatory	optional
Line Chart	1..20	1..*	no	k_2, a_1 scalar	a_2 scalar
Stacked Bar Chart	1..20	1..20	yes	a_1 scalar	-
Spider Chart	3..10	1..20	yes	a_1 scalar	-

3.3 One-Many Relationships

Relationships that are one-many (such as part_of in Figure 1) lend themselves to visualisations that are hierarchical in nature. The visualisation schema for

these relationships is illustrated in Figure 3(c), where the entity that is on the 'many' side of the relationship (such as as country for part_of) will be considered the parent entity E_p, and the other entity (province for part_of) the child entity E_c. Visualisations that represent the one-many visualisation schema are less common, but some examples are listed below together with a summary table.

- In a **tree map**, rectangles representing instances of E_p are divided into rectangles representing E_c, the area of which is proportional to the value of a scalar dimension a_1. A selector may be added to alter the proportion to be determined by other scalar dimensions $a_2, a_3, \ldots$
- In a **hierarchy tree**, nodes represent instances of E_p that are connected by lines to circles representing instances of E_c. A discrete dimension a_1 may optionally be used to colour the lines linking the entities.

One-many relationships				
Name	$\|k_1\|$	$\|k_2\|$ per k_1	mandatory	optional
Tree Map	1..100	1..100	a_1 scalar	a_2 colour
Hierarchy Tree	1..100	1..100	-	a_2 colour

3.4 Many-Many Relationships

Relationships that are many-many (such as borders in Figure 1) lend themselves to visualisations that represent networks of data. The visualisation schema pattern for these relationships is illustrated in Figure 3(d), where it should be noted that the data that governs the visualisation is now present as attributes of the relationship between entities E_1 and E_2. Visualisations that represent the many-many visualisation schema are the rarest, with two being the following:

- In **sankey** diagrams, the left hand elements of the diagram represent instances of E_1, the right hand elements represent instances of E_2, and the width of the flow between the left and right elements represents scalar dimension a_1. Optionally, a second attribute a_2 of the many-many relationship may be represented by varying the colour of the connection.
- In **chord** diagrams, instances of the entities are represented by points on the perimeter of the circle, with the value of a_1 varying the width of the connection between pairs of points. Again a second attribute a_2 of the many-many relationship may be represented by varying the colour of the connection. We note that chord diagrams are particularly suited to **reflexive** relationships, shown in Figure 3(e), since then the points around the circle represent instances of just one type of entity E, and are not grouped according to which entity type they belong to.

Many-many relationships					
Name	$\|k_1\|$	$\|k_2\|$	reflexive	mandatory	optional
Sankey	1..20	1..20	no	a_1 scalar	a_2 colour
Chord	1..100	1..100	yes	a_1 scalar	a_2 colour

4 Conclusions

We have proposed, for the first time, a conceptual modelling approach to matching data and visualisations. Our approach makes use of the conceptual schema associated with the data and automatically matches it against a set of visualisation schema patterns (expressed in the same ER formalism) each of which characterises a group of potential visualisation alternatives. We also propose the use of well-known schema transformations in order to transform the database schema to that required for matching particular visualisation patterns (details of this can be found in [3]).

With this approach, domain experts can interact with conceptual models of their data, rather than lower-level tabular representations. By providing a set of visualisation schema patterns, each of which captures the data representation capabilities of a set of common data visualisations, we make it easier for the user to select a visualisation that is meaningful in relation to their data and their information seeking requirements; and to select from a more focussed set of visualisations. By matching between the visualisation schema patterns and the conceptual database schema, full schema knowledge can be used to automatically map between the data and a range of possible visualisations. By applying, again at the level of the conceptual database schema, a set of well-known schema transformations, it is possible to generate additional matchings between the transformed database schema and the set of visualisation schema patterns.

An implementation of the approach would include also data analysis capabilities to determine whether a dimension is scalar or discrete (or both), and to determine appropriate scaling of numeric dimensions (e.g. linear, logarithmic) by supporting an additional dimension characteristic of 'skew'. Also important is extension of our visualisation schema patterns to include descriptive elements (also populated from attributes of the database schema). Finally, a full implementation would include a second stage of mapping, from a visualisation schema pattern to an actual physical visualisation representation rendered by a target data visualisation tool.

References

1. C.Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 3rd edition, 2013.
2. W. May. Information extraction and integration with Florid: The Mondial case study. Technical Report 131, Universität Freiburg, Institut für Informatik, 1999.
3. P.J. McBrien and A. Poulovassilis. Towards data visualisation based on conceptual modelling and schema transformations. Technical Report No. 39, AutoMed, 2018. www.doc.ic.ac.uk/automed.
4. X. Ren and J. Wang. Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs. *Proc. VLDB Endowment*, 8(5):617–628, 2015.
5. M. Tory and T. Moller. Rethinking visualization: A high-level taxonomy. In *Proc. Information Visualization*, pages 151–158. IEEE, 2004.