Towards Data Visualisation based on Conceptual Modelling and Schema Transformations

AutoMed Technical Report Number 39

Peter M^{c}Brien[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. The benefits of the approach are that we use the full knowledge of the conceptual model of the underlying data to identify feasible visualisations for that data; moreover, once this mapping is in place, the implementation of actual visual charts can utilise the mapping to extract data, drill-down, roll-up, pivot, switch visualisation, *etc.* 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. In this paper, 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. Section 2 gives a motivating example scenario. Section 3 introduces the notion of visualisation schema patterns each of which characterises a group of data visualisations, referring back to our motivating scenario for illustrative examples. Section 4 describes the use of common schema transformations in order to conform a dataset to a schema visualisation pattern. Section 5 discusses related work. Section 6 summarises the contributions of the paper and highlights some areas of further work.

2 Motivating Example

The latest version of the Mondial database [8] contains information that is integrated from a number of sources, and ER diagrams for two fragments of it are shown in Figure 1 and Figure 3.

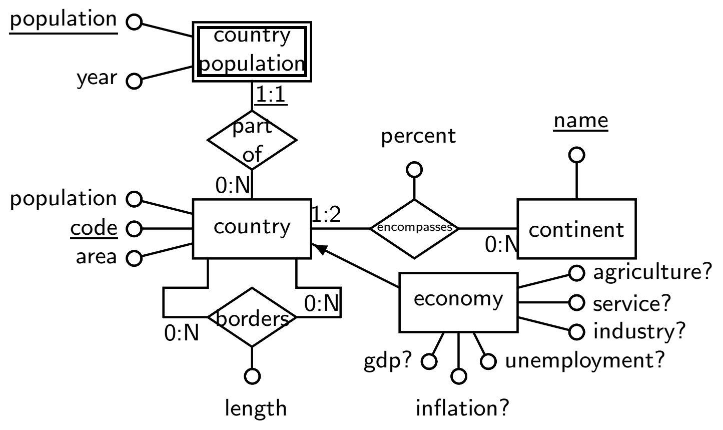
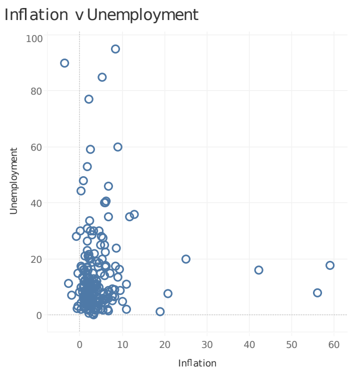


Fig. 1. ER schema of a fragment of the Mondial database

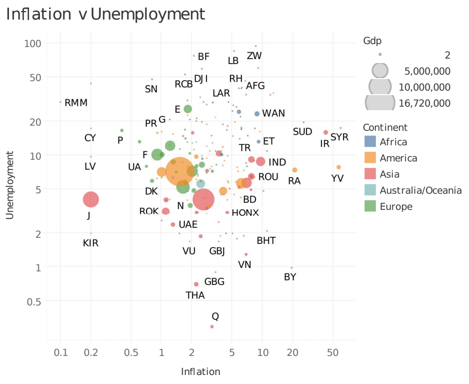
Figure 1 shows the information that is stored about countries (including the current population), and the history of a country's population — through the weak entity country_population. For some countries, data about the GDP of the country is recorded in the subset entity economy, the attributes of which 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.

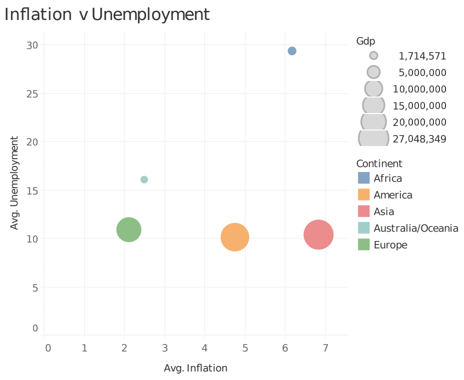
Suppose we wished to explore the relationship between inflation, unemployment, and GDP in countries. We could first extract a table of data with scheme (country, inflation, unemployment, gdp), where country corresponds to the key attribute code of the country entity in Figure 1, and without null values 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).



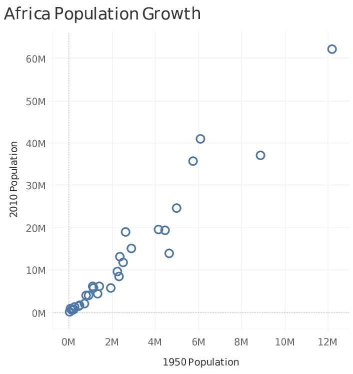
(a) Inflation v Unemployment



(b) Addition of GDP and Continent



(c) Roll-Up of data into Continents



(d) Population Growth

Fig. 2. Presentation of country data in Tableau

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 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 includes colour coding of the countries by the continent to which they belong. This is suggested by the database schema, which connects countries to continent via a relationship with restricted (upper bound 2) cardinality.

Another example of how database schema analysis can benefit the visualisation of data is shown in Figure 2(d) where data from the country_population entity for countries in Africa has been 'pivoted' to give a column for 1950 population figures and a column for 2010 figures, which then may be plotted against each other.

Our second illustrative fragment of the Mondial database is shown in Figure 3, encompassing the relationships between countries, provinces and cities, their current populations and areas, and the airports in each city.

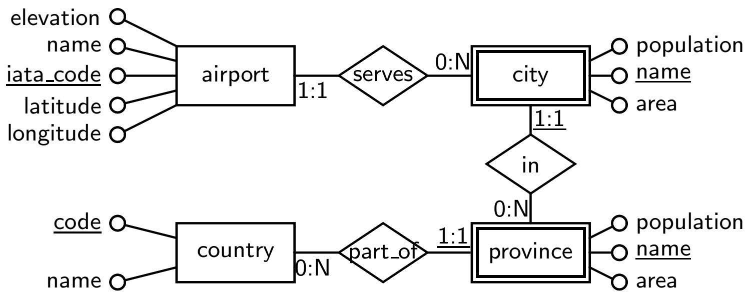
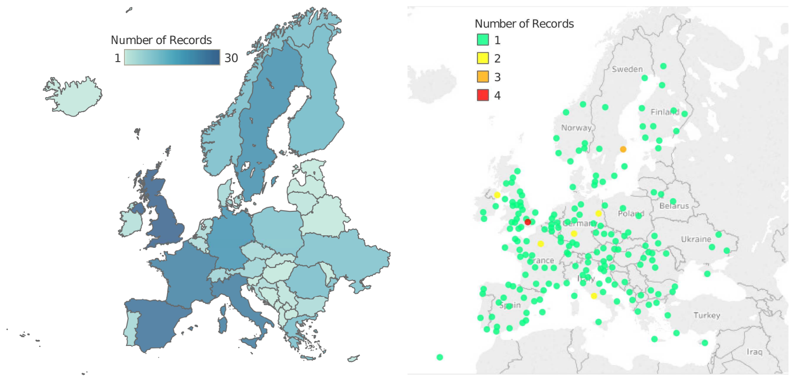


Fig. 3. Hierarchy of countries, provinces and cities

Suppose we wished to explore the relationship between airports and their locations. We could extract a table with scheme (country, province, city, iata_code), where the first three attributes are the name attributes from the corresponding entities. On presenting this to Tableau, one is able to select a map-based visualisation that shows the distribution of airports by country — see Figure 4(a). The database schema, with its one-many relationships between country, province and city, suggests that we can drill down from countries to look at airports per province, and then at airports per city — see Figure 4(b)). However, in data visualisation tools such as Tableau, this requires the user to manually select city instead of country to view the data at different levels of abstraction.



(a) Airports per Country (b) Airports per City

Fig. 4. Maps of Airports in Europe

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 [18] 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 [16]):

- **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 [18], 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.

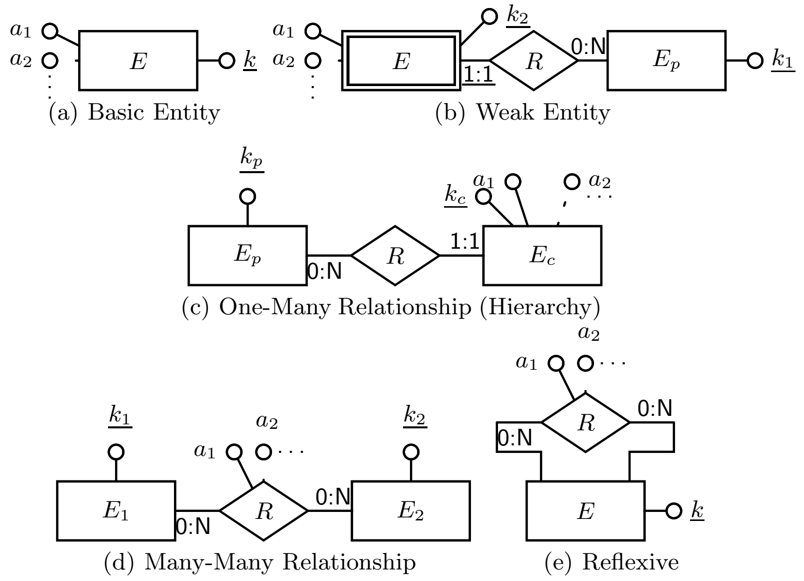


Fig. 5. Visualisation Schema Patterns for Data Visualisations

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 5(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.

- 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. Optionally, a second dimension a_2 can be used to colour the calendar entry.
- 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 the point's x and y coordinates. Optionally, a third dimension a_3 can be used to colour the point.
- 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 the coordinates of the bubble, and a third dimension a_3 its size. Optionally, a fourth dimension a_4 can be used to colour it.
- In **choropleth maps** (such as in Figure 4(a)) each region represents an instance of E, hence k must be interpretable as a geographical region, and a_1 is represented by a colour.
- Similarly in **word clouds** each word represents an instance of E and hence k must be interpretable as a lexical domain, with a_1 being the character size.

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	a_2 colour
Scatter Diagrams	1..*	a_1, a_2 scalar	a_3 colour
Bubble Charts	1..*	a_1, a_2, a_3 scalar	a_4 colour
Choropleth Maps	1..*	k geographical, a_1 colour	-
Word Clouds	1..*	k lexical, a_1 scalar	a_2 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 [11].

For example, starting with the schema in Figure 1 and matching Figure 5(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, scatter diagram, choropleth or word cloud 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 5(b), where it should be noted that k_1 would match a key relationship in the data; for example, in Figure 3 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 and city entities in Figure 3 both fail the range test, since the names of provinces are almost entirely disjoint between countries, and the names of cities are almost entirely disjoint between provinces.

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

- 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.
- A **group bar chart** is similar, except that k_2 determines the height of an element in the group. A **stacked group bar chart** combines group and bar charts to allow the weak entity key to have two parts, k_2 and k_3.
- 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.

The table below summarises the above analysis. Again the upper cardinalities shown for $|k_1|$ and $|k_2|$ are aesthetics-driven and would be user-configurable.

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	-
Grouped Bar Chart	1..20	1..20	no	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, and serves in Figure 3) lend themselves to visualisations that are hierarchical in nature. The visualisation schema for these relationships is illustrated in Figure 5(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.

- 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.
- A **circle packing** represents instances of E_p by circles, with instances of E_c placed as circles inside the circle of their parent instance of E_p. A scalar dimension a_1 is used to determine the area of the circles of E_c.

The table below summarises the above analysis.

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_1 colour
Circle Packing	1..100	1..100	a_1 scalar	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 5(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, (as found in D3, and based on the Circos package) 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 5(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.

The table below summarises the above analysis.

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 Schema Transformations to Create Visualisations

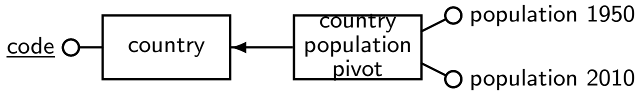
An advantage of our approach is that we can use well-known schema transformations, such as those documented in [2], to transform a database schema so that it is suitable for visualisation. We identify below three such common transformations and illustrate them through examples on the Mondial dataset. These transformations can be viewed as a way of changing the database schema so that it contains a sub-graph allowing a particular class of visualisations to be used.

4.1 Pivot

It is often the case a single graphical element needs to represent data sourced from more that one instance of an entity. For example, the chart in Figure 2(d) requires the comparison of country_population instances for a given country with year 1950 with those for the same country with year 2010. Using Datalog, we define a pivot function to handle this, as follows:

```
Pivot $E(a,b)\ \{v_1,\ldots,v_n\} \rightarrow$
  $E_p(a\_v_1,\ldots,a\_v_n)$ :-
      $E(v_1, a\_v_1)$,
      $\vdots$,
      $E(v_n, a\_v_n)$.
```

For example, Pivot country_population(year, population) {1950, 2010} gives a new entity country_population_pivot(population_1950, population_2010) that is a subset of country.



It should be noted that this is not an **information preserving** [9] transformation, since any country that does not have both a 1950 and a 2010 population figure will be missing from the pivoted country_population entity. Hence any visualisation derived from this data is not a complete representation of the 1950 and 2010 population data in country_population.[3]

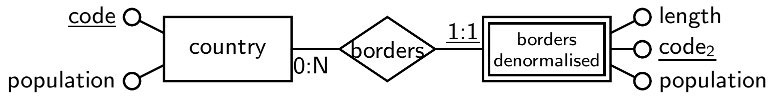
4.2 Denormalisation

A common practice is to use what we classify as one-many visualisations in Section 3 in order to represent schemas containing many-many relationships, in a kind of denormalisation of the data. The data transformation required is defined by the Datalog rule below, where starting with the many-many visualisation schema pattern in Figure 5(d), a new weak entity ED_2 is formed as a denormalised version of E_2, where the key of ED_2 is a combination of k_1 and k_2, and the attributes of R are now moved into ED_2:

```
Denormalise $E_2(k_2, x, \ldots)$ $R(k_1, k_2, a, \ldots) \rightarrow$
  $ED_2(k_1, k_2, x, \ldots, a, \ldots)$ :-
      $E_2(k_2, x, \ldots)$,
      $R(k_1, k_2, a, \ldots)$.
```

For example, applying this to the borders relationship in Figure 1:

Denormalise country(code, population) borders(code_1, code_2, length)
generates a new weak entity borders_denormalised(code_1, code_2, length, population) with length as an attribute. Note that the code_1 of the scheme appears as a relationship in the ER diagram.



When using a one-many visualisation on such a structure, using k_c as a colour dimension will ensure that the same colour is used to represent a particular value of k_c. Thus, in our example, the code_2 attribute should be used as a colour dimension.

It should be noted that this is an information preserving transformation, in that all data present in the original schema is now present in the revised schema, and hence visualisations derived from this schema can be regarded as complete representations of the original schema.

A similar rule to that given above can also be used to denormalise the data from the 'many' end of a one-many relationship into the entity representing the 'one' end of the relationship, and this may be used as a cue to present the user with the choice of roll-up or drill-down of information.

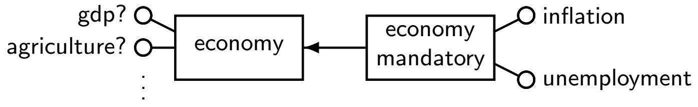
[3] It should also be noted that the physical implementation of a visualisation (such as a chord diagram) requires associations between instances of E_1 and E_2 in many-many relations to be presented with the values of E_2 forming columns of a table, and values of E_1 as rows, which can be produced by a variation of the Pivot function where tables are outer-joined together. The production of data targeted at specific visualisation tools in an area of future work.

4.3 Mandatory Attribute Specialisation

Visualisations often require that attributes in the table to be visualised have known values (not containing the value null). This means that the data needs to be reduced to include only mandatory attributes for the purpose of visualisation. The data transformation required is defined by the following Datalog rule:

```
Mandatory $E(k, a_1, \ldots, a_n) \rightarrow$
  $EM(k, a_1, \ldots, a_n)$ :-
      $E(k, a_1, \ldots, a_n)$,
      IsNotNull($a_1$),
      $\vdots$,
      IsNotNull($a_n$).
```

For example, applying Mandatory economy(code, inflation, unemployment) to the economy entity of Figure 1 will result in a subset entity economy_mandatory with two mandatory attributes, as illustrated below.



Applying this transformation to Figure 1 allows us to generate Figure 2(a). It should be noted that this transformation is not information preserving, since any instance of economy that only has one of inflation or unemployment present will be omitted from the new entity, and so Figure 2(a) will be an incomplete representation of the inflation and unemployment data.

Furthermore, the user can be guided to include additional attributes such as gdp, by producing a further specialisation of economy mandatory that includes gdp, and hence to produce the bubble chart in Figure 2(b), which is a possible basic entity visualisation in Section 3.1 that is able to model this third scalar dimension.

If the user instead wants to additionally visualise continent, they can be guided to choose a filtering of the many-many relationship encompasses that establishes a one-many relationship between continents and countries (*i.e.* to make continent become an attribute of country), *e.g.* by choosing a filtering that selects the relationship instances with percent greater than 50.

5 Related Work

The field of data visualisation is a very broad one (for reviews see e.g. [1,17,15]) and is continuing to widen with the advent of 'big data' arising from web-scale applications and the need to develop new techniques for exploring such data [5]. Current data visualisation tools (e.g. Tableau[4], D3[5], Google Charts[6]) require users to manually select data, apply transformations, and select appropriate

[4] https://www.tableau.com/products/desktop
[5] https://github.com/d3/d3/wiki/Gallery
[6] https://developers.google.com/chart/interactive/docs/examples

visual encodings from a vast array of possibilities. The user may therefore find it hard to understand the meaning of the data, the transformations that may be applied to it, and the range of visualisation possibilities, and may easily fail to 'see the wood for the trees'.

For these reasons, there has been work towards automated recommendation of visualisation possibilities and for ranking such recommendations [6,12,7,20]. The SemVis system [4] reduces the visualisation search space by using a domain ontology for mapping the source data into a visual representation ontology for storing 'knowledge about visualisation tools', and a bridging ontology to map between the domain ontology and the visual representation ontology. Our work is similar in spirit to this, but we make use of a database-specific conceptual schema that describes the domain expert's specific dataset (rather than a more general domain ontology); we also make use of the same conceptual modelling formalism (ER diagrams) to express our visualisation schemas, thus allowing common schema transformations to be applied to the former in order to conform them with the latter. Other recent work that is close to ours is the Voyager system [21] which provides a number of techniques aiming to alleviate the above problems, including faceted browsing of visualisation recommendations, and automatic clustering and ranking of visualisations according to data properties and perceptual effectiveness principles. However, this work focusses on the visualisation of a single relational table of data, and misses the opportunity to exploit a domain expert's understanding of their data at a conceptual level, and also to undertake matchings between the data and the visualisation possibilities at this conceptual level. Thus our work can be viewed as being complementary to that of Voyager.

Several works have derived taxonomies of classes of visualisation e.g. [13,3,16]. However, again these are focussed on properties of the data (dimensionality, dependent/independent variables, discrete/continuous, ordered/unordered) rather than capturing different visualisations as instances of a conceptual visualisation schema.

Finally, languages that have been proposed for manipulating graphical data (e.g. Tableau's VizQL [14], Wilkinson's Grammar of Graphics [19], R's Tidyr package [10]) require programmers to manually select data, apply transformations, and select appropriate visual encodings. For example, in the Grammar of Graphics, Wilkinson defines three algebraic operators operating on relations — **cross join**, **nest**, **blend**, while the Tidyr package supports operators such as spread for pivoting a column, gather for un-pivoting a column, split for splitting a column into two, and unite for combining two columns into one. These kinds of algebras operate at a lower-level of abstraction compared to the transformations we have illustrated in Section 4. However, such algebras could be used to implement our conceptual-level transformations *e.g.* as a target language for compilation of our conceptual-level transformations.

6 Summary and Conclusions

In this paper we propose, for the first time, a conceptual modelling approach to matching data and visualisations. We do this by making use of the conceptual schema that is associated with the data and automatically matching 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.

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.

Future work includes undertaking an exhaustive analysis of the full range of visualisations that are supported by state-of-the art tools, and extending the indicative lists given in Section 3; this analysis may give rise to additional 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. N. Andrienko, G. Andrienko, and P. Gatalsky. Exploratory spatio-temporal visualization: an analytical review. *Journal of Visual Languages & Computing*, 14(6):503–541, 2003.
2. C. Batini and M. Lenzerini. A methodology for data schema integration in the entity relationship model. *Trans. Software Engineering*, 10(6):650–664, 1984.
3. S.K. Card and J. Mackinlay. The structure of the information visualization design space. In *Proc. Information Visualization*, pages 92–99. IEEE, 1997.
4. O. Gilson, N. Silva, P.W. Grant, and M. Chen. From web data to visualization via ontology mapping. 27(3):959–966, 2008.

5. E.Y. Gorodov and V.V. Gubarev. Analytical review of data visualization methods in application to big data. *Journal of Electrical and Computer Engineering*, 2013:22, 2013.
6. J.Mackinlay. Automating the design of graphical presentations of relational information. *Trans. On Graphics*, 5(2):110–141, 1986.
7. J. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. *Trans. Visualization and Computer Graphics*, 13(6), 2007.
8. W. May. Information extraction and integration with Florid: The Mondial case study. Technical Report 131, Universität Freiburg, Institut für Informatik, 1999.
9. R.J. Miller, Y.E. Ioannidis, and R. Ramakrishnan. Schema equivalence in heterogeneous systems: Bridging theory and practice. *Inf. Systems*, 19(1):3–31, 1994.
10. R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2013.
11. X. Ren and J. Wang. Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs. *Proc. VLDB Endowment*, 8(5):617–628, 2015.
12. S.F. Roth, J. Kolojejchick, J. Mattis, and J. Goldstein. Interactive graphic design using automatic presentation knowledge. In *Proc. CHI*, pages 112–117. ACM, 1994.
13. B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The Craft of Information Visualization*, pages 364–371. 2003.
14. C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *Trans. Visualization and Computer Graphics*, 8(1):52–65, 2002.
15. A.C. Telea. *Data visualization: principles and practice*. CRC Press, 2014.
16. M. Tory and T. Moller. Rethinking visualization: A high-level taxonomy. In *Proc. Information Visualization*, pages 151–158. IEEE, 2004.
17. M.O. Ward, G. Grinstein, and D. Keim. *Interactive data visualization: foundations, techniques, and applications*. CRC Press, 2010.
18. C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 3rd edition, 2013.
19. L. Wilkinson. *The Grammar of Graphics*. Springer, 2005.
20. G. Wills and L. Wilkinson. Autovis: automatic visualization. *Information Visualization*, 9(1):47–69, 2010.
21. K. Wongsuphasawat *et al*. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *Trans. Visualization and Computer Graphics*, 22(1):649–658, 2016.