

the system, environment, and/or user from a set of observations as captured via events and/or data. Inference can be employed to identify a specific context or action, or can generate a probability distribution over states, for example. The inference can be probabilistic—that is, the computation of a probability distribution over states of interest based on a consideration of data and events. Inference can also refer to techniques employed for composing higher-level events from a set of events and/or data. Such inference results in the construction of new events or actions from a set of observed events and/or stored event data, whether or not the events are correlated in close temporal proximity, and whether the events and data come from one or several event and data sources.

[0020] Referring now to **FIG. 1**, there is illustrated a general block diagram of a system of the present invention. The present invention provides the capability of allowing a user to work with a representative structure of a relational database **100** even though remote therefrom and disconnected. Such a scenario is common in that the user may be denied access rights and/or privileges to the relational database **100** or even to a network on which the database is disposed.

[0021] The disclosed architecture addresses the development of a Relational Schema Definition (RSD) language component **102** that generates an RSD file **104** that represents the complete structure and data of the relational database from which it is derived.

[0022] The RSD file **104** can then be made accessible to a user instead of the database **100** itself. This facilitates the user working with the database **100** indirectly via the RSD file **104** while traveling or in any scenario where the user is disconnected from the database **100** (e.g., the database is offline). The location of the RSD file **104** may be anywhere, e.g., in this embodiment, local to the relational database **100**, such that the user can be given access to it. Of course, the user may be required to login to the network and/or the database **100** to gain access to the file **104**, or may be given free access to the file **104**. This implementation is at the discretion of the user.

[0023] The disclosed RSD language format is based upon an XML (eXtensible Markup Language) that is used to represent the relational schema. However, as indicated hereinabove, in lieu of XML, the relational schema can be represented with an alternative declarative language. The RSD component **102** includes a declarative, implementation-neutral format such that after relational database metadata is obtained, the RSD file **104** can easily be generated, stored, and used by applications to regenerate the relational schema of the database **100**. Thus, the disclosed architecture facilitates use of the RSD file **104** in a remote and disconnected environment such that a user can take the RSD file **104** offline, and use the file **104** to regenerate the relational database **100** in its entirety for processing, instead of having to maintain a connection to the relational database **100** in order to access its contents.

[0024] Databases and XML offer complementary functionality for storing data. Databases store data for efficient retrieval, whereas XML offers an easy information exchange that enables interoperability between applications. To take advantage of XML features, database tables can be converted into XML documents. XML tools can be employed

with such documents for further processing. XML documents can be presented as, for example, HTML (HyperText Markup Language) pages with XSLT (Extensible Stylesheet Language Transformation) stylesheets, can be searched with XML-based query languages such as XQuery (XML Query Language), can be used as a data-exchange format, and so on. For processing XML documents, XML tools can work with any suitable API, e.g., a DOM API ((Document Object Module Application Programming Interface). Thus, XML tools can be used to treat databases as if they were XML documents. This way, the need to convert a database is obviated.

[0025] It is preferable to have a file in a non-procedural declarative format that describes the schema of the relational database that is understandable by applications. It is written in the XML format and XML syntax, and consequently, is easy to parse, easy to load into an XML parsing API (e.g., DOM), and easy to understand.

[0026] The RSD language **102** also facilitates moving (or mapping) data between the relational database **100** and an Object component **106** and/or an XML component **108** using a mapping component **104**. This is accommodated by using a declarative means rather than a conventional procedural mechanism (e.g., executing C++ code against a result set abstraction to generate an object or an XML structure/component). The capability to map data from one data model to a different data model is a desirable operation in great demand with data environments of today. That is, data environments that are diverse, and employ a wide range of mechanisms and mediums for persisting and accessing data. With respect to Object data, XML data, and Relational databases, the means to map data between each of these different data structures is important, since users are continually modifying their data storage schemas, mediums, and processes.

[0027] Thus, there is provided the relational database **100** having a relational schema therein represented in the form of metadata, and from which the metadata can be retrieved by the RSD component **102** for generating the RSD file **104**. The RSD language component **102** prepares the database data for mapping to another data model via the mapping component **104**. The mapping component **104** can then map the data to at least the Object component **106** and/or the XML component **108**. Note, however, that the RSD component **102** can be used in conjunction with the mapping component **104** to map relational data to an arbitrary domain.

[0028] The use of an Object Schema Definition (OSD) language component **1112** to process Object data **114** for use by the XML component **108** and a Relational component **116**, and an XSD language component **118** to process XML data **120** for use by the Object component **106** and the Relational component **1116**, are not part of this description, as indicated by dotted lines. Note that the particular source data (**100**, **114**, and **120**) and the associated language (**102**, **112**, and **118**) are not restricted to data transformation to a different target component (**106**, **108**, and **116**). That is, a scenario can exist where the source relational database **110** can be mapped to a target relational database (the relational component **116**) having a different relational structure. Thus, the RSD language component **102** and mapping component **110** facilitate this process. Similarly, a scenario can exist