

where the source object data **114** can be mapped to a target object data model (the object component **106**) having a different object structure. The OSD language component **112** and mapping component **110** facilitate this process. Still further, a scenario can exist where the source XML data **120** can be mapped to a target XML data model (the XML component **108**) having a different XML structure. The XSD language component **118** and mapping component **110** facilitate this process.

[0029] Referring now to FIG. 2, there is illustrated a flow chart of a process for generating RSD from the relational database. While, for purposes of simplicity of explanation, the one or more methodologies shown herein, e.g., in the form of flow chart, are shown and described as a series of acts, it is to be understood and appreciated that the present invention is not limited by the order of acts, as some acts may, in accordance with the present invention, occur in different orders and/or concurrently with other acts from that shown and described herein. For example, those skilled in the art will understand and appreciate that a methodology could alternatively be represented as a series of interrelated states or events, such as in a state diagram. Moreover, not all illustrated acts may be required to implement a methodology in accordance with the present invention.

[0030] Flow begins at **200** where a tool of the disclosed architecture is activated according to trigger data. Some types of triggering events are described hereinbelow. At **202**, the tool, as part of the RSD component, executes to walk through the relational database metadata to find the tables and columns and, relationships therebetween. At **204**, the tool provides the capability to allow the user to select all or a subset of the relationships for use in the RSD file. At **206**, the tool allows the user to make a selection. At **208**, the tool creates the RSD file of the selected relationships that precisely describe the database structure and data. At this point, optional extensions are included to support implementation-specific extensions and derivations from an ANSI (American National Standards Institute) standard schema (e.g., SQL Server, Oracle . . .). At **210**, the RSD file is stored for later access. The process then reaches a Stop block.

[0031] Referring now to FIG. 3, there is illustrated a general block diagram of the RSD language component **102** of FIG. 1. The RSD component **102** includes a tool **300** for extracting the metadata from the relational database **100** and using the metadata to generate the RSD file **104**. The overall schema of the RSD file **104** is a combination of physical information **302** (or elements) and logical information **304** (or elements) used to describe the relational database **100**. The physical information **302** can be harvested directly from the database **100** automatically using the tool **300**, while user annotations to that information are added incrementally to provide the logical element thereof. Annotations are made based upon user knowledge of the relationship between the database tables. This can be performed manually or automatically. Where automatically, a smart algorithm can be employed to derive the annotation information from, e.g., foreign key constraints. Thus the smart algorithm can walk through the database extracting this information for annotating the table relationships. This can be further automated by requiring a degree of certainty that the annotations are correct. Thus, if the user requires that the automated annotation process achieve a minimum 95% accuracy, performance less than this may require manual correction and

review to ensure the database is precisely captured in the RSD file **104**. This process can also be performed via a classification process that is described in greater detail hereinbelow.

[0032] A database is typically defined at least according to tables and columns. The relationships between tables is not well-defined, which are the logical components of the relational database. The logical elements are useful for representing the semantics of the database, for mapping the database to another data model, for modeling, etc. A way to obtain a "hint" at the table relationships is via a foreign key. In order to describe a relationship between tables, the logical element is used. Thus, given an RSD file, the relational database is recreated by using both the physical and logical elements. The tool **300** is sufficiently sophisticated to handle merge scenarios where the RSD file **104** has been updated by the user with logical information and is then refreshed from the database **100**.

[0033] The following convention is adopted such that element names that are written in Times **12** plain text font are denoted as physical elements that are derived directly from the database **100**. These elements are appropriately regenerated from the database **100** each time the generation tool is run. Element names that are written in Times **12** italics text font are denoted as logical elements that can be annotated with an IsLogical attribute. If the IsLogical attribute is TRUE, RSD generation tools respect the user-supplied extensions and do not overwrite that information when updating the RSD file **104** from the database **100**.

[0034] The RSD file generation process can be initiated manually or automatically. Manual operation simply requires that the user initiate the process by way of a user interface or other communication means. When performed automatically, the tool **300** can be triggered to operate according to any number of trigger mechanisms. The RSD file generation process can be initiated according to predetermined time criteria (e.g., hourly, daily, weekly) to process the current state of the database. Thus, the RSD file **104** could be updated every ten minutes by running the tool to extract the latest state of the database. However, this fixed time increment may not provide that latest state of the database if the database is updated after the most recent RSD file generation.

[0035] Alternatively, the tool can be automatically activated to generate an updated RSD file **104** after a database change has been detected. For example, if it is determined that five percent of the database **100** has experienced changes, the tool **300** could be automatically triggered to update the RSD file **104**.

[0036] Still alternatively, the tool can be activated to generate the RSD file **104** only when the database is accessed by a non-relational database query, either before the query is made, or after the query is completed. However, this too may involve more time than is desirable, since the requestor may then need to wait until the process complete, if performed before the query.

[0037] In another scenario, if certain portions of the database **100** are determined to be a higher priority data than other portions, then after changes have been made in the higher priority data, the tool could be automatically activated to update the RSD file **104**.