Proceedings of the SURVEYING & SPATIAL SCIENCES BIENNIAL CONFERENCE 2011
21-25 November 2011, Wellington, New Zealand

Schema Element Dependencies in a Federated Spatial Database System

Xiaoying Wu; Jianhong (Cecilia) Xia; Geoff West; Bert Veenendaal
Department of Spatial Sciences, Curtin University
Kent St, Bentley, WA, 6102, Australia
wxy_1975@hotmail.com;c.xia@curtin.edu.au; g.west@curtin.edu.au; b.veenendaal@curtin.edu.au

Lesley Arnold
Landgate
1 Midland Square, Midland WA 6056, Australia
lesley.arnold@landgate.wa.gov.au

ABSTRACT

A federated spatial database is the integration of multiple spatial data sources and the realisation of effective spatial data sharing. However, in a federated database environment, database schemas are subject to change.

Some schema elements depend on other schema elements and therefore schema changes result in dependent schema elements becoming invalid in a federated spatial database. Schema element dependencies are more complex in a federated database environment because dependencies include both intra-dependencies (within a database) and inter-dependencies (between databases). This makes management of schema evolution in a federated environment far more difficult than traditional systems.

Schema element dependency (SED) is an important component of managing schema evolution. This paper systematically reviews the schema element dependencies in a federated spatial database system, proposes a SED metamodel to identify schema element dependencies and a SED metadata schema to store schema element dependencies metadata, and explains the methods and steps to generate SED metadata. This paper also presents a SED tool that is used to generate SED metadata and conduct schema change impact analysis. SED provides a foundation for schema evolution management in a federated spatial database.

KEYWORDS: Database schema changes; Federated Spatial Database System; Schema Element Dependency; Schema Element Dependency Metamodel; Impact Analysis

1 INTRODUCTION

A database schema describes the structure of the database. A spatial database schema is an extension of the traditional schema and includes spatial types, indexes and functions (Yeung & Hall, 2007). Spatial database schemas, like traditional schemas, are subject to change due to changes in representation of reality and application requirements. Changes can also occur as a consequence of integration with other systems, compliance to new regulations and the implementation of new security requirements.

Schema change can invalidate the applications that are based on the schema through both the changes in a schema element itself and the dependencies of the changed element. That is because a change of one database schema element might also affect other schema elements and result in other schema elements being invalid. The task of managing schema evolution is to ensure the consistency and integrity of data and application after change.

A Federated Spatial Database System (FSDBS) integrates various geographically distributed spatial data sources and provides a unified data access mechanism that facilitates spatial data sharing (Yeung & Hall, 2007). In a FSDBS, spatial data are shared by multiple organisations and applications across a number of databases. This has been achieved through advancements in networks and communications, distributed computing technologies, and conformance with standards and policies. FSDBS has been an active research subject in dealing with database heterogeneity (Litwin et al., 1990) and can be seen as an important part of Spatial Data Infrastructure (SDI).

A basic FSDBS consists of (Yeung & Hall, 2007): (1) A Federated Spatial Database Management System - an ordinary DBMS that includes a server, a federated database and the system catalogue. It communicates with applications, receives requests from applications, analyses queries and discomposes them into subqueries, sends subqueries to relevant data sources and composes results from data sources then sends back the end users and application. The federated database here is a virtual database which integrates multiple spatial data sources. System catalogue contains schema information of the federated database as well as matching and mapping information between schemas; (2) Spatial data sources which are autonomous and heterogeneous; and,(3) Applications. Applications here could be desktop applications, and, web applications. For example, the federated server can work with a web feature server to provide web mapping services. Figure1 illustrates the basic architecture of a FSDBS.

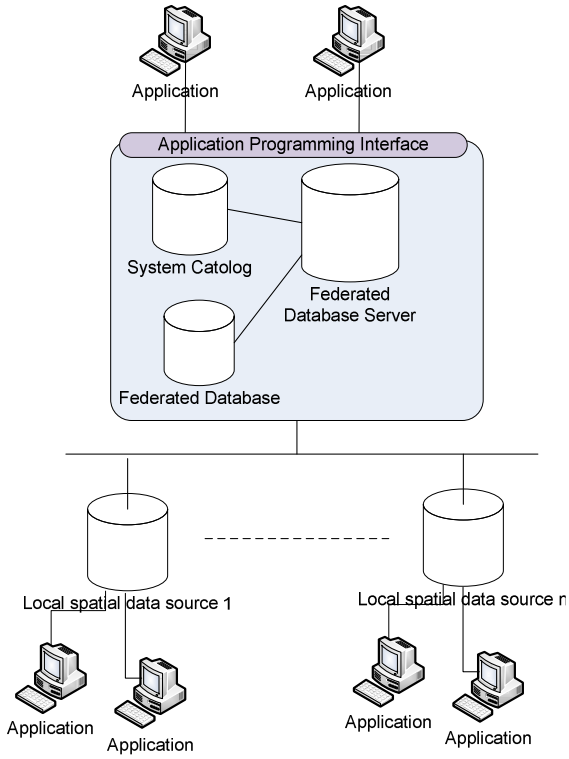

Figure1. Architecture of A FSDBS

In a FSDBS, a local database schema change may affect not only the database itself but also other databases involved in the system. Applications are affected by both local schema changes well as changes elsewhere in the federated schema. This is because the federated schema is dependent on local schemas. Managing schema evolution in a FSDBS environment overcomes the mismatch between applications and the evolved schemas, and maintains consistency of the correspondences between the local schemas and the federated schema before and after schema changes.

Schema Element Dependency (SED) details the dependencies between attributes in the various relations in the databases in a FSDBS. It is important for managing schema evolution. It can help to conduct schema change impact analysis, and to identify affected schema elements after schema changes so as to modify them to adapt to the new schema. Schema element dependencies are more complex in the FSDBS environment as dependencies in such a system

include both inner-dependencies (within a database) and inter-dependencies (between databases). Therefore, management of schema evolution in this system is more challenging.

Currently schema management is performed manually and there is a need to develop methods that can be used to deal with schema evolution automatically or semi-automatically. This research has developed such methodologies, and the conceptual Automatic Schema Evolution (ASE) framework to manage schema evolution in a federated spatial database system in an automatic manner. This paper focuses on Schema SED in the ASE which includes the concept of SED and its functions, SED Metamodel, SED schema in the metadata repository, the methodologies to generate SEDs, as well as the SED tool developed.

2 BACKGROUND

This section briefly introduces some concepts and background information in relation to schema element dependencies.

2.1 Architecture of Automatic Schema Evolution

An Automatic Schema Evolution framework has been developed for managing schema evolution in a FSDBS (Wu et al., 2011b). Figure 2 illustrates the architecture of the ASE. There are five main components included in the architecture: spatial databases, a metadata repository, a view rewriting tool, a schema mapping tool and a schema element dependency tool (Wu et al., 2011a).

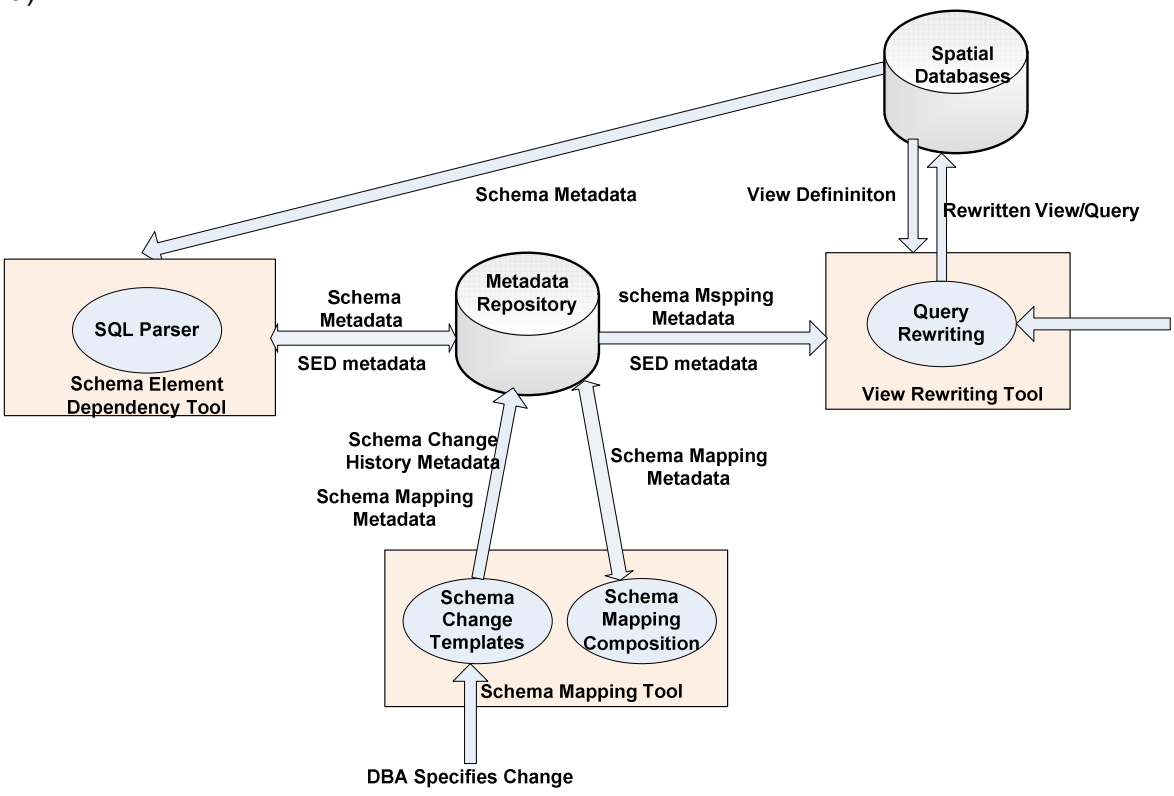

Figure 2. The ASE Architecture

Spatial databases are all databases involved in the FSDBS that include all local spatial data sources and the federated or virtual spatial database as illustrated in Figure 1.

The Metadata Repository is the central repository that stores various types of metadata. Metadata is data about data (Sen, 2004). In this research, metadata is treated as the first class for schema evolution management. The metadata repository provides a consistent and united access mechanism to improve the effectiveness of schema evolution management. Four types of metadata are included: (i) schema metadata; (ii) schema element dependency metadata; (iii) schema mapping metadata; and (iv) schema change history metadata.

The Schema Mapping Tool includes two modules: Schema Change Templates (SCTs) and Schema Mapping Composition. The functionality of this tool is to generate and update schema mapping metadata which then can be used for view/query rewriting.

The SED tool is used to generate SED metadata in the FSDBS, and to update the SED metadata when a schema changes. Based on the SED metadata generated by the SED tool, the schema change impact can be analysed and affected views can be identified when database schema changes.

The View Rewriting Tool that includes the query rewriting module is used to rewrite views against the old schemas into views against the new schema as determined by the schema mapping metadata.

2.2 Spatial Schemas

In order to model the spatial aspects of real world objects, different approaches have been proposed and developed. For example, Modeling Application Data with Spatio-temporal features model (MADS) defines constructs and a language to describe an application schema including thematic and spatial-temporal data structures, and multi-representation aspects (Parent et al., 2006). Spatial object types supported in MADS include both feature and coverage types. ISO 19109 - Rules for application Schema (2005) defines a general feature model to describe the feature object types and their properties including attributes, operations and associations. A Universe of Disclosure described by the constructs and modelling languages defined in MADS or ISO 19101 is a conceptual spatial schema that is platform independent. The defined spatial data types including points, lines and polygons can represent spatial attributes in the conceptual spatial schemas.

With the development of database technologies, spatial object types and their properties, described by the conceptual spatial schema, can be implemented in a relational database. For example, the implementation of feature object types is supported by contemporary DBMSs. The Simple Features Access (SFA) specification (defined in ISO 19125) defines the Structured Query Language (SQL) implementation of the geometry object model that is consistent with ISO 19109 - Rules for application Schema (2005). SFA specifies the SQL schema for storage, query, retrieval and update of spatial data via a SQL interface. SQL implementations include the implementation of primitive data types including the binary data type, and support for extended geometry data types (OGC, 2006a, 2006b). However, coverage objects are not included in the SFA specification so storage, query and management of coverage data are not supported by the SQL interface.

The relationship between the conceptual spatial schema and the SQL spatial schema is explained in Section 3.2: SED Metamodel.

2.3 View and Spatial View

In a traditional database, a view is defined as a stored query, usually as SQL statements. Views can be used to provide a flexible representation of a database by providing users with only data of interest and concealing the rest (Date, 2003). A spatial view extends the traditional view by including a spatial field. A spatial view (or map view) is a live window to the underlying geographic database (Arnold, 2005). Such spatial views are created in the same way as traditional non-spatial views and stored as spatial queries that contain spatial attributes and operations such as spatial selection and spatial join.

2.4 Schema Metadata

In a database, schema metadata is stored in the system catalogue. According to the SQL standard, there are Definition Schema base tables and Information Schema views stored in the database catalogue (ISO/IEC, 2006). The Definition Schema provides a data model for the Information Schema. Examples of Definition Schema tables include *TABLES* and *COLUMNS*. The *TABLES* table stores all tables including views and the *COLUMNS* table has one row for each column. Information Schema views “are viewed tables defined in terms of the base tables of the Definition Schema”(ISO/IEC, 2006). Information Schema views can be accessed the same way as

other tables in the database. Users can query the views if they are granted the privilege. Examples of Information Schema views include *TABLES* view and *COLUMN* view (ISO/IEC, 2006).

In commercial DBMSs, system views are often used to display metadata. For example, in SQL Server, system catalogue views are most widely used for obtaining metadata that is used by the SQL Server Database Engine. All database schema metadata can be derived through catalogue views (MSDN, 2009).

In a federated database, the system catalogue stores both database schema metadata and mapping metadata between the federated database schema and local database schemas.

3 SCHEMA ELEMENT DEPENDENCY

3.1 Schema Element Dependency

Conventionally, a SQL schema consists of schema objects like tables, views, columns, keys and constraints in a relational database (Stephens & Plew, 2001). Some schema objects refer to others when they are defined. For example, a view refers to one or more tables and views. Normally, the object that is being referred to is called a referenced object. The one which references other objects is called a dependent object (Oracle, 2011). The dependency can occur between different types of schema objects such as table to table, view to table, view to view, and trigger to table etc.

If one schema of an object is changed, the dependent objects may become invalid. For example, a change to a base table can affect the view when a view references the base table: (1) If the base table is renamed or dropped, the view will become invalid; (2) if the columns in the base referenced by the view are renamed or dropped, the view will become invalid too; (3) however, any changes on the columns that are not referenced by the view will have no impact on the view. Dependency information at the level of schema objects, however, can't provide a clear answer to the impact of schema changes. In order to better manage schema evolution, dependencies of interest in the research are taken to be column level dependencies which are on a finer level.

In order to represent column level dependencies, the concept of schema element has been introduced in this paper. Schema elements include both schema objects and columns. Schema elements of interest in the research are mainly tables, views and columns. With schema elements, column dependencies and view dependencies can be represented.

A column dependency represents the dependency between two columns. For example, a computed column is derived from other column(s) in the same table by an expression and is a virtual column. Therefore, the computed column is dependent on the columns in the expression. Column dependencies also occur when one column in a table references the primary key or candidate key in another (or the same) table via a foreign key, also called referential integrity.

In a spatial database, a view (traditional and spatial view) is defined in the form of a SQL statement. Dependency between a view and a column of a table/view occurs when the query defined in the view references the column of the table/view. Dependencies between a view and columns of tables/views are called view dependencies in this paper.

In a federated database system, schema element dependencies occur not only within a database, but also across databases. Since the federated database is a virtual database based on other data sources, tables in the federated database refer to tables in other databases.

Finding all the dependencies in a database system is very important in terms of minimising error, avoiding missing information and ensuring consistency in the database. With accurate and detailed schema element dependency information, a schema change impact analysis can be conducted before schema changes; and affected schema elements can be identified and then modified accordingly to adapt to the changed schema.

3.2 SED Metamodel

The SED metamodel (illustrated in Figure 3 using UML syntax) is used to represent column level schema element dependencies in a FSDBS. This conceptual metamodel consists of three

parts: the schema elements of a conceptual spatial schema, the schema elements of a SQL schema, and the SEDs.

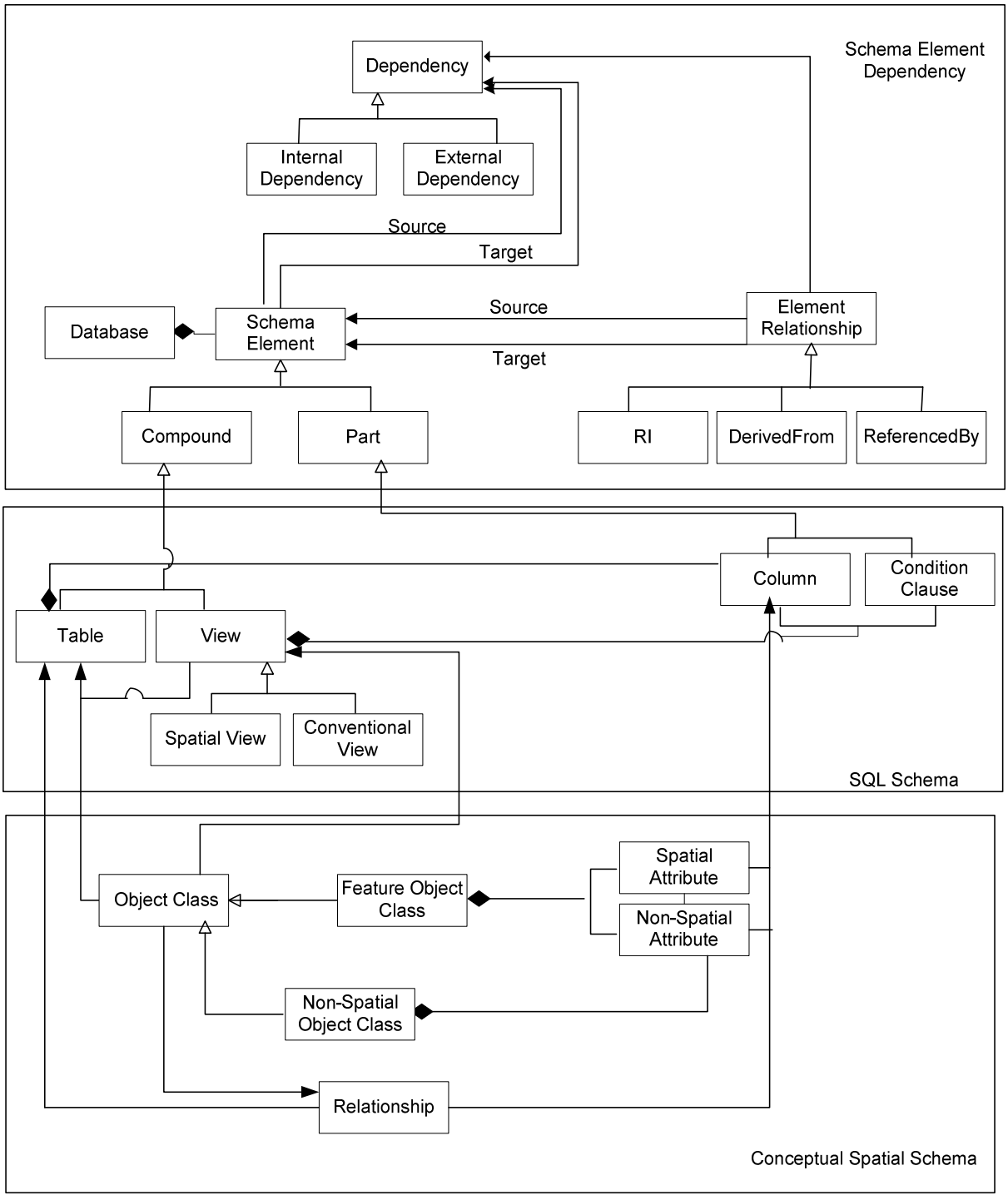

Figure 3. SED Meta-model

The conceptual spatial schema is a conceptual data model adapted from the generic feature model from ISO 19109 (2005). Schema elements defined in the model are object classes consisting of feature and non-spatial object classes, attributes including spatial and non-spatial attributes, and relationships. A feature object class represents a collection of vector objects of the same type. A non-spatial object class represents a category of objects with no spatial characteristics. An object class has a name and attributes including spatial (for a spatial object class) and non-spatial attributes. Each attribute has its own properties including name and data

type etc. Spatial attribute data types are points, lines or polygons. Object classes can have relationships between them.

The SQL schema is an implementation of a conceptual spatial schema according to SFA specifications (OGC, 2006a, 2006b). Schema elements include tables and views that are supported by contemporary relational or object-relational DBMSs. A table contains the name of the table, names of all columns and data types for these columns. A view includes the name of the view and names of columns. These are required because a view is a virtual table that is derived from base tables or views by means of stored queries.

There are three types of table in a DBMS: feature table, geometry table and non-spatial table. A feature table stores a collection of features of the same type. Attributes of a feature object class including the spatial attributes are columns of a feature table. A geometry table stores the geometric information of the geometric objects. Tables without any spatial element are called non-spatial tables.

The relationship between the conceptual spatial schema and the SQL schema can be seen in Figure 3. Object classes are stored as tables in the relational database. More specifically, the implementation of a non-spatial object class is a non-spatial table while the implementation of a feature object class varies depending on the data types supported by the underlying DBMS. For example, a feature object class uses a feature table and a geometry table, with a key reference between them if the predefined data types in the DBMS are used. However, only the feature table is needed if geometry data types are supported by the DBMS. Attributes of an object class are stored as columns of the corresponding table. A relationship between two object classes is implemented as either a table or the primary key and foreign key column of the two tables.

Schema element dependencies consist of schema elements and the relationships between them. Schema elements are divided into *Compound* and *Part*. The *Part* component denotes atomic elements such as a column. The *Compound* component represents the elements that contain parts such as tables and views. Relationships include *Referential Integrity (RI), DerivedFrom* and *ReferencedBy*. A relationship connects two schema elements. The dependency information, termed a column level dependency, can be derived from the relationships between the source and the target elements that are connected by the relationship. Depending on the databases containing the source and target elements, the dependency will be identified as an internal or external dependency.

3.3 Schema and SED Metadata Schema

As mentioned in the background section, there are four types of metadata stored in the metadata repository: (i) schema metadata; (ii) schema element dependency metadata; (iii) schema mapping metadata; and (iv) schema change history metadata. SED metadata in the metadata repository, as shown in Figure 2, is used for schema change impact analysis before schema changes and detection of affected schema elements after schema changes. Figure 4 is the schema of database schema metadata and SED metadata.

Among the seven tables shown in Figure 4, five tables are used to store database schema metadata: SED_Server, SED_Database, SED_Owner, SED_Tabview and SED_Column. Based on the names of these tables in the schema, it is not hard to understand what metadata is stored in each table: SED_server stores information on all servers involved in the FSDBS; SED_Database is for information on all spatial databases participating in the FSDBS; SED_Owner is used to store details of the owners of tables/views in each database; SED_Tabview is for all tables and views; and, SED_Column is for all columns in each table/view.

The other two tables in Figure 4: SED_Viewd and SED_Columnd are used to store different types of SED metadata. The SED_Viewd table stores dependencies between views and columns of tables/views. In this table, the dependent element is the view while the referenced element is the column in a table/view. The SED_Columnd table stores dependencies between columns generated by computed columns or referential integrity. In SED_Columnd, the dependent and referenced elements are both columns and these columns may be in the same table or in different tables.

Schema metadata can be extracted from the original database system catalogue. The generation of SED metadata varies depending on the type of dependency. Details of the generation of SEDs are explained in section 4: Generation of SEDs.

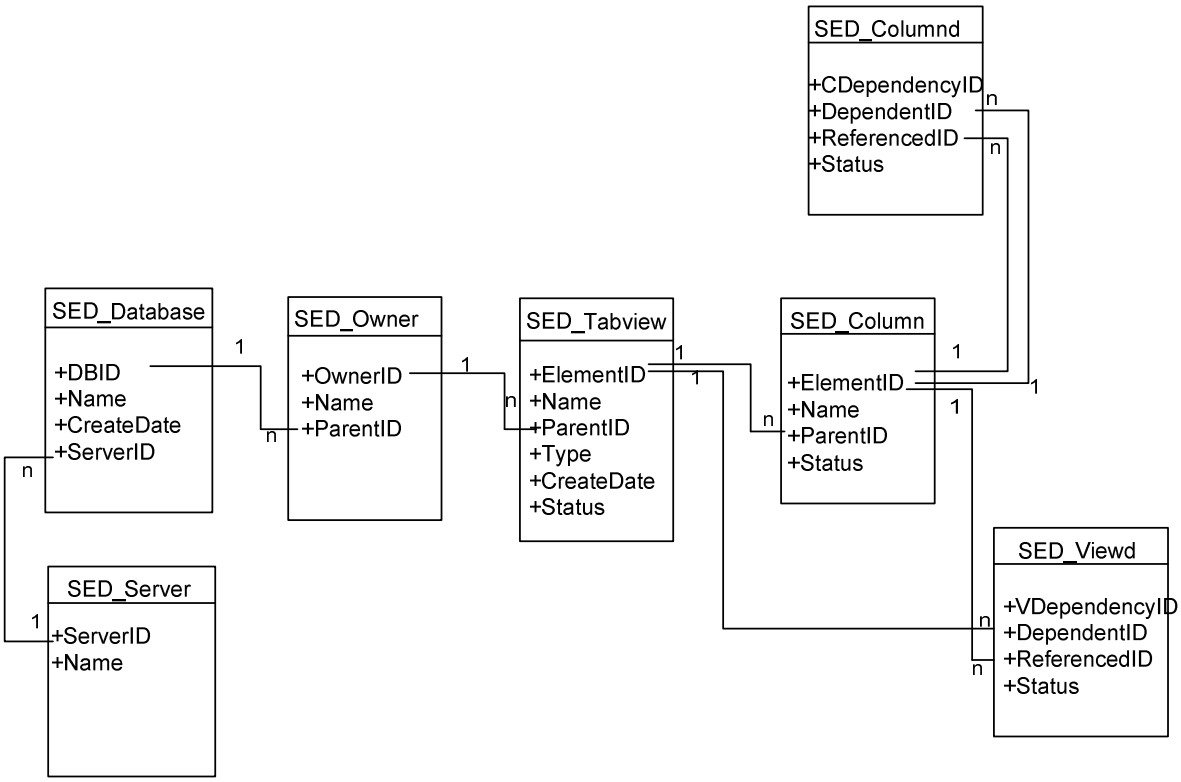

Figure 4. Schema and SED Schema

4 GENERATION OF SEDS

Different approaches have been used to generate different types of dependencies.

4.1 View Dependencies

To generate dependencies between a view and base tables (or views) the view definition is analysed. View definitions are SQL statements stored in the system catalogue that can be extracted using a SQL statement.

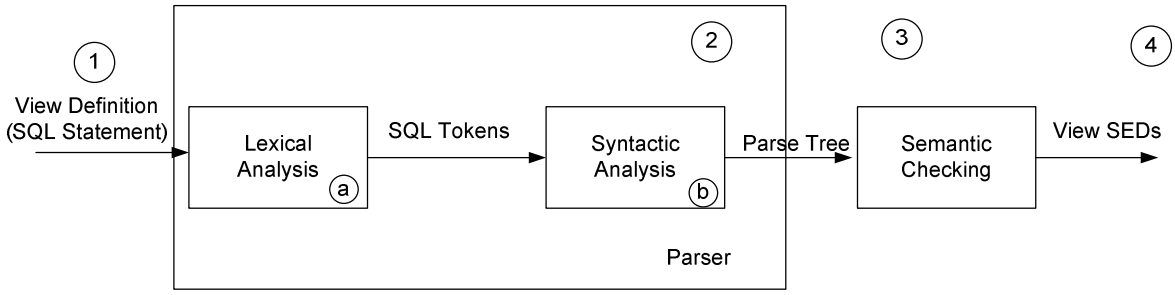

Figure 5. Steps to generate View SEDs

The steps to generate view SEDs are shown in Figure 5:

1. View metadata extraction: View definitions are stored in the system catalogue and a view can be derived by a SQL statement.
2. Parsing the view definition SQL statement to generate a parse tree. Two stages are involved in this step: lexical analysis and syntactic analysis:
    a) Lexical analysis is the process of breaking text into a set of tokens. The input of this stage is the text written in a certain language, and, the output is a set of tokens. In SQL, tokens include ordinary and delimiter tokens. Ordinary tokens include numeric constants, ordinary identifiers, host identifiers and key words. Delimiter tokens include string constants, delimited identifiers, operator symbols and other special characters (IBM, 2011). Examples of key words are SELECT and FROM. Identifiers are schema elements such as tables, columns, databases. After tokenisation of the SQL statement a sequence of SQL tokens is formed, which are the input of the syntactic analysis.
    b) Syntactic analysis is the process to generate a grammatical structure (normally called a parse tree) based on the tokens formed by lexical analysis according to the grammar of the language.
3. Semantic checking. The parse tree is the syntactic representation of the view definition. In order to generate view SEDs, semantic checking has to be conducted on the parse tree. This can be done by consulting the database schema metadata to determine the relationships between schema elements and hence the dependency relationships between schema elements.
4. Insert the view SEDs information in the SED-Viewd table in the metadata repository.

View dependencies generated with this approach include both internal and external database dependencies. A federated database is a virtual database that can be seen as views built against component databases. Therefore, dependencies between schema elements in the federated database and component databases can be similarly generated.

This approach can also be extended to generate dependencies caused by triggers, procedures or function since they are mainly composed of SQL statements. They are outside the scope in this paper.

4.2 Column Dependencies

Column dependency occurs when there is a computed column or referential integrity. Steps to generate column dependencies caused by a computed column is similar to view dependencies and include:(1) extraction of the definition of the computed column from the system catalogue; (2) analysis of the definition according to the SQL standard and extract schema elements in the definition; and (3) insert the dependent element (the computed column) and referenced elements (extracted from the definition) into the SED_Column table in the metadata repository.

The generation of column dependencies caused by integrity is relatively straightforward. It can be achieved by extracting the dependency data from the system catalogue and inserting it into the SED_Columnd in the metadata repository.

4.3 SED tool

In order to generate and update SEDs automatically, a SED tool has been developed in the .NET framework using C# based on the methodologies described above. The functions of the tool include, as shown in Figure 6: (1) extraction of schema metadata from the system catalogue and insert into metadata repository; (2) generation or update of SEDs including view SEDs and column SEDs; and (3) analysis of schema change impact.

The SED tool can be used to extract different ranges of schema metadata depending on what is specified in the interface. For example, if a database is specified, all schema metadata in the database can be extracted, and if a table is specified, only schema metadata related to the table will be extracted. By these means, the schema metadata in the metadata repository can be synchronised with the related spatial databases.

The main function of the tool is to generate SEDs metadata, as indicated by the name of the tool. This is mainly achieved by the SQL parser imbedded with the tool, as shown in Figure 5. The SQL parser is included to parse the SQL statements such as the view definition and column definition. SEDs can be generated when a new schema element is created and updated when a schema element is changed. For example, when a new view is created, the SEDs for the view can be generated by specifying the view in the user interface.

With the SED metadata stored in the metadata repository, the SED tool can also be used to conduct impact analysis of schema changes when the changing schema element is specified. For example, when a column of a table is specified in the interface, all schema elements that depend on the column will be displayed.

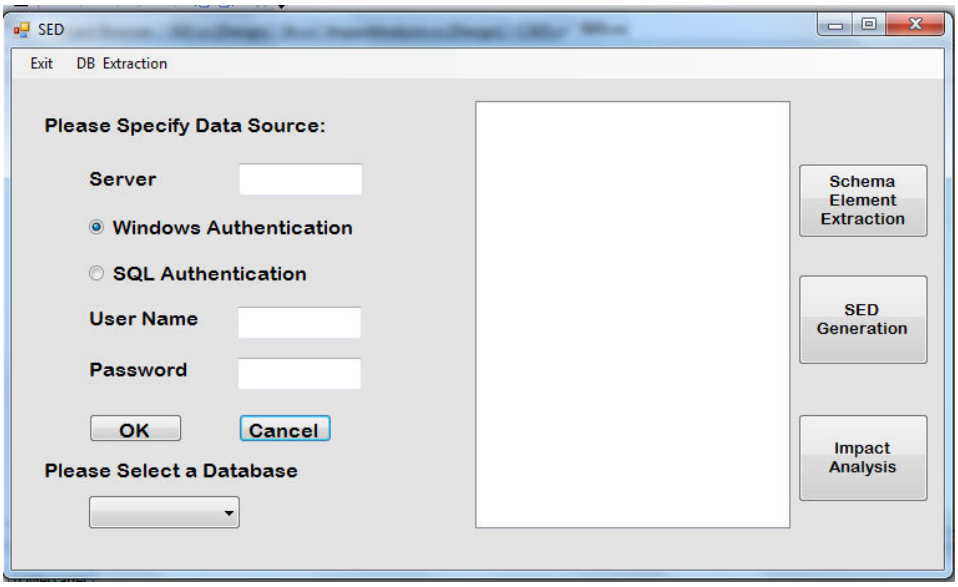

Figure 6. SED Tool

4.4 Implementation of SED tool in a FSDBS

A test environment involving a sample federated spatial database system has been set up to test the SED tool. The system comprises a federated ArcGIS geodatabase G and two component geodatabases S1 and S2. Figure 7 illustrates the schemas of geodatabases involved in the sample system. The underlying relational DBMS used is SQL Server 2008. S1 consists of two feature object classes - *ExistingMainRoads* and *ProposedMainRoads*, and one non-spatial object class – *RoadType.* A View Highway displays all highways including both existing and proposed. S2 consists of one feature object class – *LocalRoads* and one non-spatial object class – *FeatureNames*. The federated schema G includes one feature object class – *RoadSegments* that displays all road segments including both main roads and local roads.

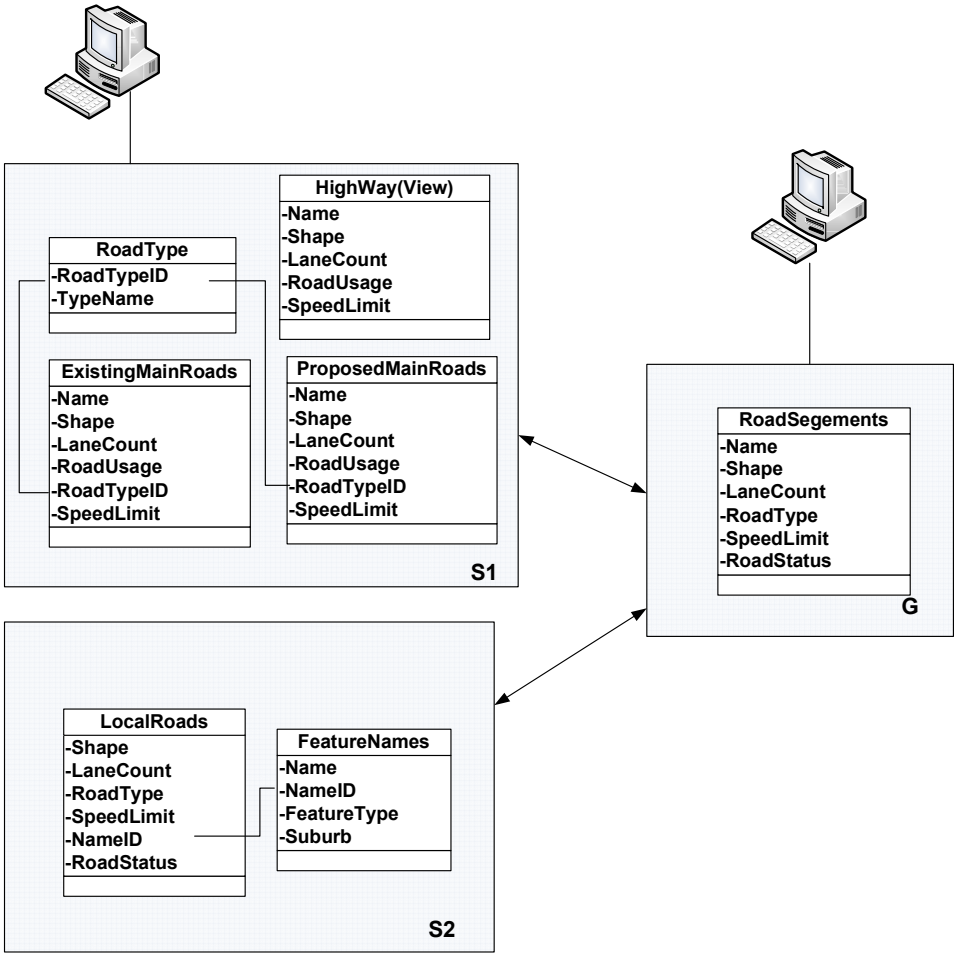

Figure 7. Schema of the Sample Federated Spatial Database System

The tool extracted schema elements from the two local spatial databases and the federated spatial database, generated view dependencies and column dependencies including both inner-dependencies and inter-dependencies, and detected the affected elements if a schema element is specified. Snapshots of view dependencies and column dependencies generated by the tool are shown in Figure 8 and Figure 9 respectively. Figure 8 shows the dependencies for each view. For example, the first 14 rows are the dependencies of the view “Highway” on the columns of other tables, and the first two rows mean the view is dependent on column “LaneCount” from table “PROPOSEDMAINROADS” and “EXISTINGMAINROADS”. Figure 9 shows the columns dependencies e.g. the first row shows there is a dependency of column “RoadTypeID” of table “PROPOSEDMAINROADS” on column “RoadTypeID” of table “ROADTYPE”.

DependentView	Downer	Ddatabase	Dserver	ReferenedColumn	RTabview	Rowner	Rdatabase	Rserver
Highway	dbo	S1	GIS\ARCSDE_SQL	LaneCount	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	LaneCount	EXISTINGMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	Name	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	Name	EXISTINGMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	RoadTypeID	ROADTYPE	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	RoadTypeID	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	RoadTypeID	EXISTINGMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	RoadUsage	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	RoadUsage	EXISTINGMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	SHAPE	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	SHAPE	EXISTINGMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	SpeedLimit	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	SpeedLimit	EXISTINGMAINROADS	dbo	S1	GIS\ARCSDE_SQL
Highway	dbo	S1	GIS\ARCSDE_SQL	TypeName	ROADTYPE	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	LaneCount	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	Name	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	RoadTypeID	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	RoadUsage	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	SHAPE	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	SpeedLimit	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	RoadTypeID	ROADTYPE	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	TypeName	ROADTYPE	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	LaneCount	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	Name	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	RoadTypeID	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	RoadUsage	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	SHAPE	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	SpeedLimit	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	Name	FEATURENAMES	dbo	S2	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	NameID	FEATURENAMES	dbo	S2	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	LaneCount	LOCALROADS	dbo	S2	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	NameID	LOCALROADS	dbo	S2	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	RoadStatus	LOCALROADS	dbo	S2	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	RoadType	LOCALROADS	dbo	S2	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	SHAPE	LOCALROADS	dbo	S2	GIS\ARCSDE_SQL
RoadSegment	dbo	G	GIS\ARCSDE_SQL	SpeedLimit	LOCALROADS	dbo	S2	GIS\ARCSDE_SQL

Figure 8. View Dependencies in the Sample FSDBS

DColumn	Dtabview	Downer	Ddatabase	Dserver	Rcolumn	Rtabview	Rowner	Rdatabase	Rserver
RoadTypeID	PROPOSEDMAINROADS	dbo	S1	GIS\ARCSDE_SQL	RoadTypeID	ROADTYPE	dbo	S1	GIS\ARCSDE_SQL
RoadTypeID	EXISTINGMAINROADS	dbo	S1	GIS\ARCSDE_SQL	RoadTypeID	ROADTYPE	dbo	S1	GIS\ARCSDE_SQL
NameID	LOCALROADS	dbo	S2	GIS\ARCSDE_SQL	NameID	FEATURENAMES	dbo	S2	GIS\ARCSDE_SQL

Figure 9. Column Dependencies in the Sample FSDBS

5 CONCLUSIONS

Schema evolution is essential in a FSDBS because it is dependent on databases owned, maintained and, importantly, modified by other organisations. Effective management of schema evolution is an integral part of a SDI. It ensures that discovery and access to spatial data and services is consistent even though schema changes have occurred.

Managing schema evolution in such an environment is a significant challenge. To manage this effectively, it is important to identify all existing schema element dependencies and to maintain this over the lifetime of schema changes. This paper discusses the SEDs including the concept, types and function of SED and methodologies to generate SEDs in a FSDBS. A SED tool has also been developed to extract and synchronise schema metadata, generate and update SEDs, and analyse schema change impact and detect affected schema elements when schema changes.

With the SED metadata generated, further work can be conducted on the components of the Schema Mapping Tool and View Rewriting Tool, as illustrated in Figure 2, which includes developing schema change templates, generating and storing schema mapping metadata, schema mapping composition, and developing tool to rewrite views. These aspects combined will further automate and enhance the management of schema evolution in a FSDBS.

ACKNOWLEDGEMENT

The authors would like to thank Landgate and Curtin University for providing funds to support this PhD research.

REFERENCES

Arnold, L. (2005). *Dynamic Spatial Updating: A Scale-independent Approach to Data Management and Thematic Map Revision*. PhD Research, Curtin University of technology.

Date, C. J. (2003). *An Introduction to Database System - 8th Edition*: Addison Wesley.

IBM. (2011). Tokens Retrieved 03/07, 2011, from http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.sqlref/rch2tok.htm

ISO. (2005). ISO 19109 *Geographic information -- Rules for application schema*

ISO/IEC. (2006). Information technology - Database Languages - SQL - Part 11: Information and Definition Schemas (SQL/Schemata).

Litwin, W., Mark, L., & Roussoupoulos, N. (1990). Interoperability of Multiple Autonomous Databases. *ACM Computing Surveys, 22*(3), 267-293.

MSDN. (2009). Catalog Views (Transact-SQL) Retrieved 14/01, 2010, from http://msdn.microsoft.com/en-us/library/ms174365.aspx

OGC. (2006a). OpenGIS® Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture. Retrieved from http://portal.opengeospatial.org/files/?artifact_id=25355

OGC. (2006b). OpenGIS® Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option. Retrieved from http://portal.opengeospatial.org/files/?artifact_id=25354

Oracle. (2011). Oracle® Database Concepts --11g Release 1 (11.1) Retrieved 10/05, 2011, from http://download.oracle.com/docs/cd/B28359_01/server.111/b28318/dependencies.htm#i1234

Parent, C., Spaccapietra, S., & Zimányi, E. (2006). Conceptual Modeling for Traditional and Spatio-Temporal Applications, the MADS Approach. *Springer*.

Sen, A. (2004). Metadata management: past, present and future. *Decision Support Systems, 37*(1), 151-173.

Stephens, R., & Plew, R. (2001). *Database Desgin*: Sams Publishing.

Wu, X., Xia, J., West, G., Arnold, L., & Veenendaal, B. (2011a). *An Architecture of Managing Schema Evolution in a Federated Spatial Database System*. Paper presented at the 7th International Symposium on Digital Earth, Perth.

Wu, X., Xia, J., West, G., Arnold, L., & Veenendaal, B. (2011b). Managing Schema Evolution in a Federated Spatial Database System *Discovery of Geospatial Resources: Methodologies, Technologies and, Emergent Applications*: IGI Global.

Yeung, A. K. W., & Hall, G. B. (2007). *Spatial Database Systems - Design, Implementation and Project Management* (Vol. 87): Springer.