Table of Contents

Figures

Chapter 3: SDP Conceptual Data Reference Model Overview

In This Chapter

- Learn about the purpose of a Conceptual Data Reference Model.
- Understand the methodology used to generate the SDP Conceptual Data Reference Model.
- Discover the business rules used to implement the Conceptual Data Reference Model as a physical or XML Schema representation.
- Learn about the differences among conceptual, logical, physical (entity-relationship) and XML Schema models.

Schedule Data Profile
The Schedule Data Profile (SDP) is a specification that describes operator generated schedule and related data. It is a business semantics specification that describes schedule information, specifically each data element and its relationship to scheduling data concepts, and preserves the referential integrity of these data concepts.

The SDP will be based on recognized information technology (IT) standards such as Extensible Markup Language (XML) and XML Schema, as well as standards and best practices in the IT and transit industries.

Introduction to This Chapter

Although the two basic formats that are needed to understand the Guidance document are the SDP CDRM and the SDP XML Schema, there are other methods that may be used to implement the SDP Conceptual Data Reference Model. This chapter describes the purpose of the CDRM with respect to its *reference* representation as well as the methods used to implement it.

Purpose of a Conceptual Data Reference Model

The purpose of a Conceptual Data Reference Model (CDRM) is to describe the "real-world domain" using unambiguously defined set of data concepts and model their relationship to each other. The technical language and methodologies used in the development of a CDRM can be confusing, in part, because of different uses of the same terminology.

Basically, the SDP's CDRM is an abstract model that describes the real world domain of transit scheduling. The technical approach or methodology used included entities, relationships and attributes, which are captured in a model that is independent of technology implementation. In other words, the CDRM can guide more than one way to implement the sharing of transit schedule data. For example, a database management system and SDP XML document are two methods that may be used to implement the SDP CDRM.

The effort may be likened to building a house in the Tudor style. We know we want a Tudor style house, that is, "steeply pitched roofs, half-timbering often infilled with herringbone brickwork, tall mullioned windows, high chimneys, jettied (overhanging) first floors above pillared porches, dormer windows supported by consoles."[1] We also need to develop a framework that supports both the style and the types of features needed by the family that will live there (number of rooms, heating system, etc). The prospective house may be modeled in a number of different ways, such as in a three-dimensional form, or a series of diagrams from multiple perspectives. Finally, we need to develop a set of blueprints and identify the specific

[1] From http://en.wikipedia.org/wiki/Tudorbethan_architecture.

materials that will be used to build the house. One set of materials may necessitate changes to the blueprint or impact other types of materials specified to be used to build the house.

The different information technology modeling approaches used to develop the SDP are no different. The SDP must be described in several different ways in order to support the fundamental needs and goals of the Schedule Data Profile. In this section, we describe the different modeling technologies and their role with respect to the final product, that is, the Schedule Data Profile XML document of schedule data submitted by each Agency.

The CDRM was developed to help ensure that the needs of the applications that use schedule data are met by the SDP. It identified key requirements related to the data validity including: identity, uniqueness, references, attributes, and data format and type. These requirements were captured in the SDP Functional Requirements and modeled in the SDP CDRM. As a result, the model helps ensures that data, when implemented and subsequently exchanged, are consistent and well understood across applications.

The SDP's CDRM accomplishes the following:

- Provides a reference for the meaning and relationships of the real-world domain concepts when they are implemented in applications and interfaces;
- Defines the identifiers and uniqueness requirements required for downstream applications;
- Provides a description of core attributes that are needed by most downstream applications;
- Incorporates flexibility to constrain or extend the model given various implementation approaches and tools.

Methodology for SDP Reference Model Development

The SDP Project used a system engineering approach for developing user driven requirements. The initial stage of the process involved soliciting stakeholder input on how key user groups currently use and might in the future use schedule and related information. This *Concept of Operations* (ConOps) was developed through stakeholder meetings and interviews. A set of downstream application operational descriptions, high level requirements, and detailed data requirements were documented in a series of white papers. The white papers (also called Use Cases) included downstream applications such as Integrated Trip Planning, Dynamic Generation and Presentation of Public Timetables, and Generation of Ad Hoc Scheduling.

The High Level requirements from the Use Cases provided the initial input into the scope and requirements for the conceptual reference model. A draft CDRM was developed, and through a series of stakeholder meetings, the model was refined in order to better capture user upstream practices and downstream application needs. The product from this effort was a comprehensive Functional Requirements document. The requirements document not only covered the schedule and related data concepts found in the reference model, but it also described a preliminary set of requirements for naming and formatting an exchange standard based on XML Schema standard, as well as integration issues for a central repository that fuses individual agency data into a regionally related data set.

The CDRM is meant to be used as a framework to unambiguously describe the SDP data concepts and their relationship to each other. Different technical methods may be used to

physically represent and store schedule data. The three methods include: logical data model, physical database, and XML Schema. The first two models (logical and physical) are shown using an entity-relationship notation. Appendix D describes the symbols and conventions of the notation in more detail.

Differences Between the SDP CDRM and Implementation Methods

The SDP CDRM uses an entity-relationship (ER) method to represent real-world phenomena. The model is driven by a set of requirements described by current, local practice and by best practices advocated by the information technology and transit industries. The CDRM uses an abstract ERD modeling method to represent these real-world phenomena. Although a similar notation is used to describe the Logical and Physical models, they do not include the same information or serve the same purpose. Differences between the CDRM and Logical, CDRM and Physical and CDRM and XML Schema models are described below.

Difference between a CDRM and SDP Logical Entity Relationship Model: A CDRM shows the relationship between entities, but does not carry related keys to related entities. For example, in a system that supports more than one schedule version per agency, the schedule version identifier must be included in every entity in the logical model. A *logical model (expressed as an ER diagram (ERD))* shows these primary and foreign keys, and thus describes key storage requirements related to the data set. The CDRM makes no assumptions about how a model is applied, rather, it describes real-world relationships.

Difference between a CDRM and SDP Physical Implementation: Similar to the relationship between a conceptual and logical model, the physical implementation supports primary and foreign keys, the procedures that validate these relationships, and specific formats and syntax related to each data type described in the model. Specifically, these rules and procedures are defined for a specific database management system such as Oracle 9i, MS Access 2003, etc.

Difference between a CDRM and SDP XML Schema Implementation: The SDP XML Schema's primary purpose is to facilitate the sharing of data across different information systems, particularly via the Internet. The SDP XML Schema uses the CDRM to describe schedule and related data concepts and preserve the relationship requirements among data concepts for one schedule version and for a single transit operator. A set of rules were used to migrate the data concepts from the CDRM to the XML Schema implementation. The general set of rules are described below, detailed rules pertaining to each entity are documented in the following chapters.

Rules for Implementing the SDP XML Schema from the CDRM

The specific rules for migrating the CDRM to the SDP XML Schema implementation include the following:

- Each entity became a complex type. Some entities that define types of things, like day type, pattern type became an enumerated type or code embedded in the complex type.
- An entity that is related to another entity and typically acquires a foreign key in a logical model, may be dealt with in one of two ways.

  - The entity that is related to another entity is embedded as a child element in another element. For example, RouteDirection is embedded as element <routeDirectionList> in the RouteStructure.
  - The complexType description for the entity carries an element similar to a foreign key. For example, Trip includes elements routeID and patternID.
- If a related entity becomes a stand-alone element and carries a related key (like Trip includes patternID and routeID), then the element value is constrained using the "keyref" notation (as described in the XML Schema standard).
- Even if there is reason to embed an element into a parent, the XML Schema should not include more than four layers, that implies that the maximum number of children is less than four elements from the root node.

An Example of Migrating the CDRM to a Logical, Physical and XML Schema Representation

As described above, there are rules for implementing the CDRM from the conceptual framework to its logical, physical and XML schema formats. The following sections show how the CDRM for the same data concept, Schedule Calendar Date, is transformed to a logical, physical, and XML Schema model. Each model depicted in Figures 3-1 through 3-7, is summarized in the list below:

- Conceptual Data Reference Model—*Figure 3-1*
  - Note the CDRM, Logical and Physical models are all represented using ERD notation.
- Logical ERD Model—*Figures 3-2 and 3-3*
  - Note the key identifiers become primary keys (pk), and related entities include related or foreign keys (fk);
- Physical Model—*Figure 3-4*
  - Note the attributes are specified with specific data types that reference the specific database management system specifications;
- XML Schema—*Figures 3-5 to 3-7*
  - Note the data are formatted in a hierarchical format, and relationships must be maintained by internal functions.

Example of the Conceptual Data Reference Model

The CDRM is expressed as an Entity Relationship Diagram (ERD). Figure 3-1 shows an example for the basic representation of the schedule calendar date concept.

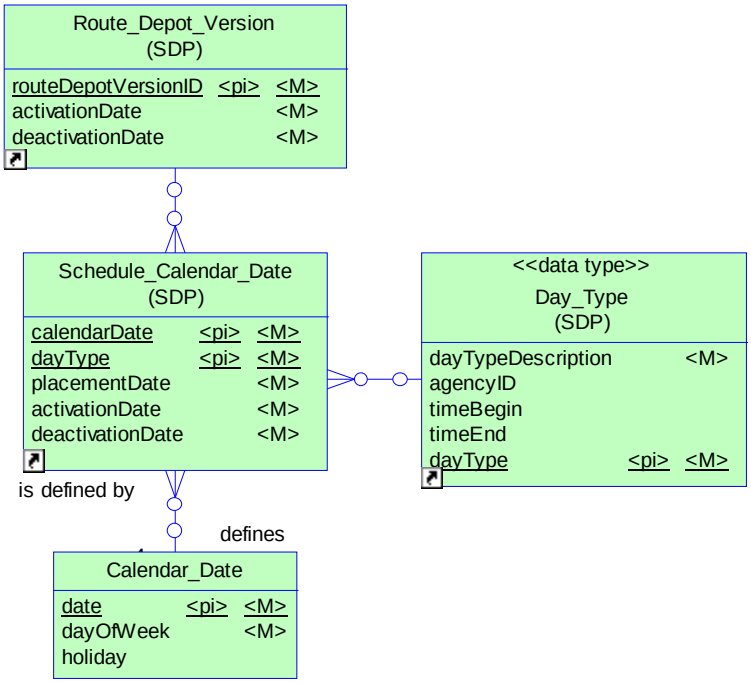

Figure 3-1: Conceptual ER Model of Schedule Calendar Date Concept

The following paragraph describes the requirements of Figure 3-1:

> "Transit service is scheduled for each day of operation. Service components may be scheduled to operate on different dates depending on a number of factors. These factors may be schedule based; for example, special trips are designated when there is an event at Shea Stadium or service to evacuate workers from the city during a snow storm. The Schedule Calendar Date associates the relevant schedule components (designated by the Route Depot Version) and an index related to the appropriate trips (designated by the day type) into a table which is used as a reference.
>
> "A Schedule Calendar Date is created for each set of schedule version components and the trips that operate on the specific dayType. In some cases, the schedule version components are scheduled for only part of a day, for example, the schedule components vary when the Mets play games that begin at 5 p.m. versus at 7 p.m." [*SDP Functional Requirements*, p. 102]

Example of the Logical Entity-Relationship Representation

The logical model is driven by application requirements related to how the data are stored and accessed. In the example illustrated in Figure 3-2, the Schedule Calendar Date entity inherits related keys designating the schedule version when more than one schedule version is present. When an organization changes its schedule mid-version, the entity is required to include a revision number, and when a transit agency issues their schedule by route or by route and depot, the route-depot version is also included in the entity. When this model is extended to a regional repository, each entity must designate the authority that issued the data, as such, the functional entities Route_Depot_Version, Schedule_Calendar_Date and Day_Type include the agency

identifier (agencyID). The actual implementation of the conceptual to logical model may be seen in Figure 3-6.

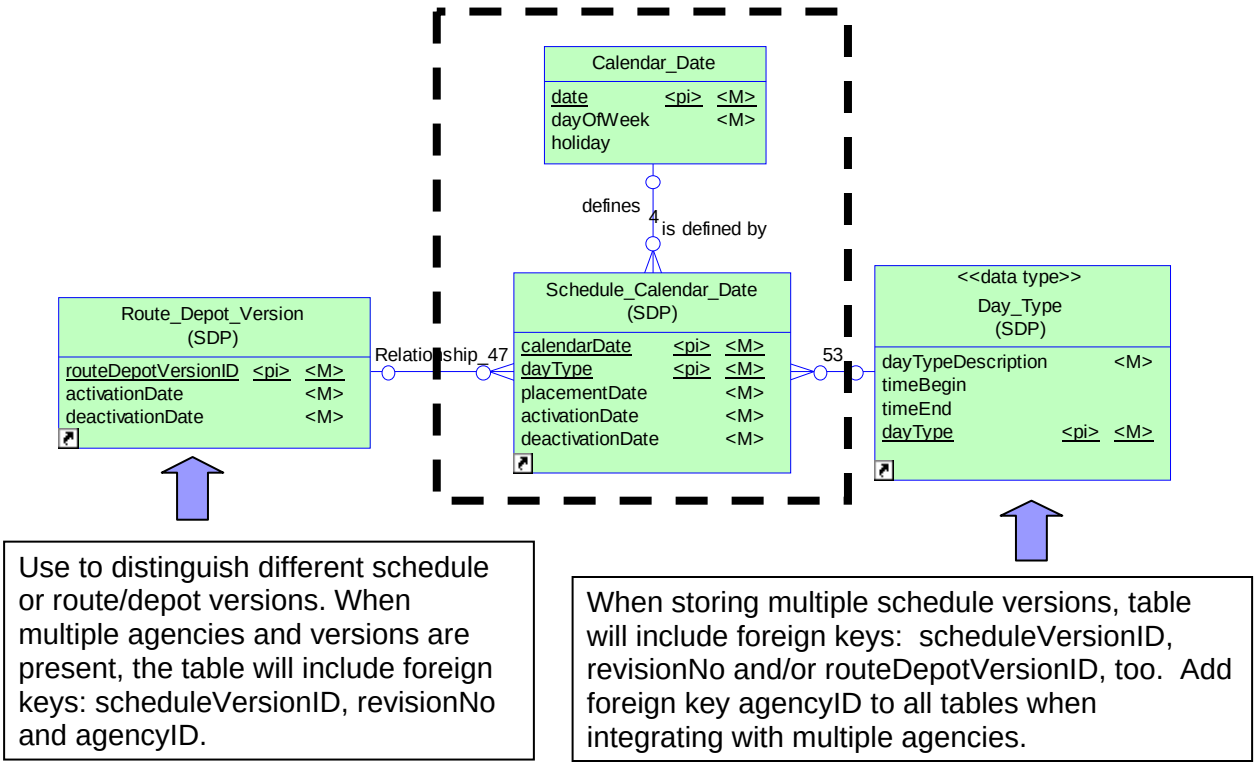

Figure 3-2: Migrating From Conceptual to Logical Model

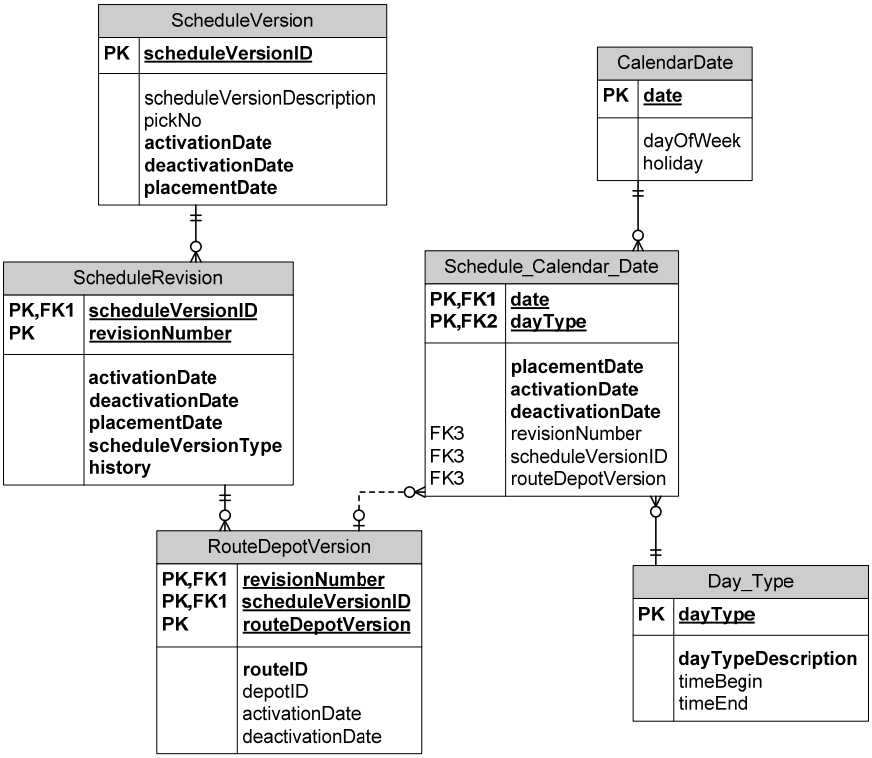

Figure 3-3: Logical Model of Schedule Calendar Date Concept

Example of the Physical Database Implementation

The physical model is similar to the logical model except the data types are defined by the database management system. The physical database also supports procedures that enforce referential integrity triggers (primary and foreign keys) when data are added, changed or deleted from the database. A generic physical model for the Schedule Calendar Date concept is illustrated in Figure 3-4. As is shown in the figure, this is similar to the logical model shown in Figure 3-3 **except** for defining specific data types and showing the procedures. For organizations with specific database management systems, the logical and physical representations are somewhat redundant since a logical and physical models will use the same data type definitions.

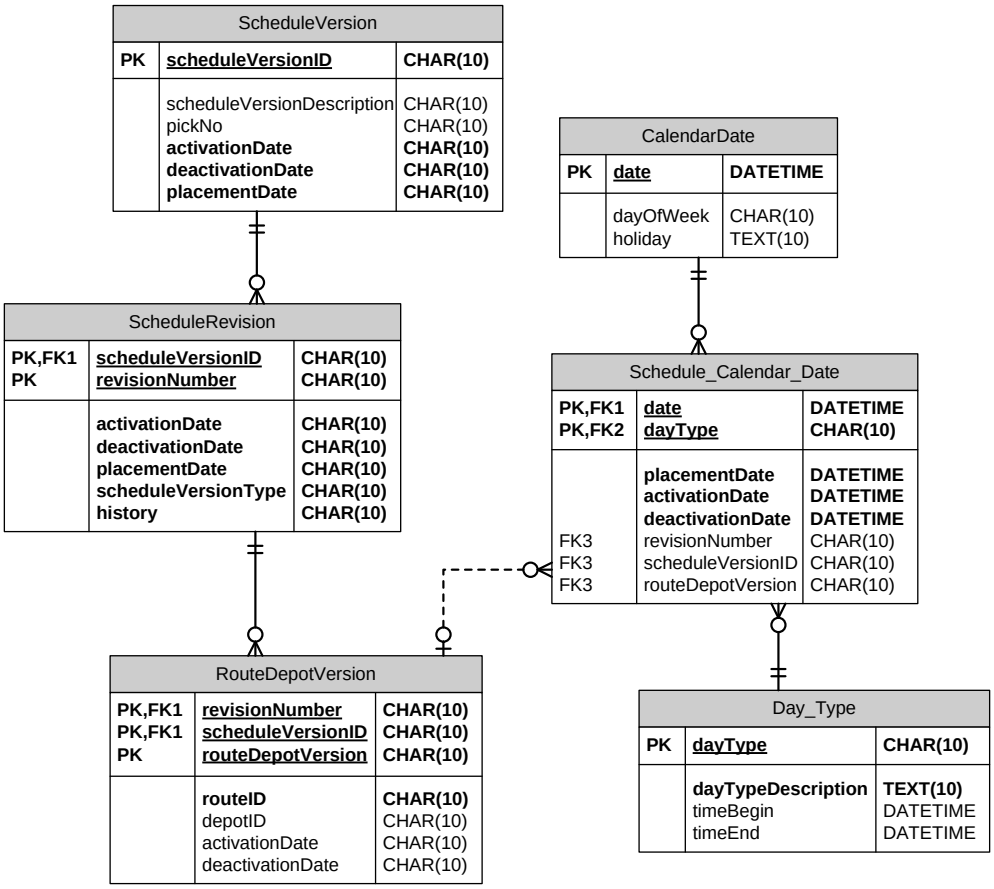

Figure 3-4: Physical Model of the Schedule Calendar Date

Example of the XML Schema Implementation

An XML Schema is organized like a hierarchical database. In the case of the special Schedule Calendar Date, the schema reflects two key elements (since the schedule version, revision and route depot version elements are defined in the SDP schema)[2]. The two elements are depicted in Figure 3-5.

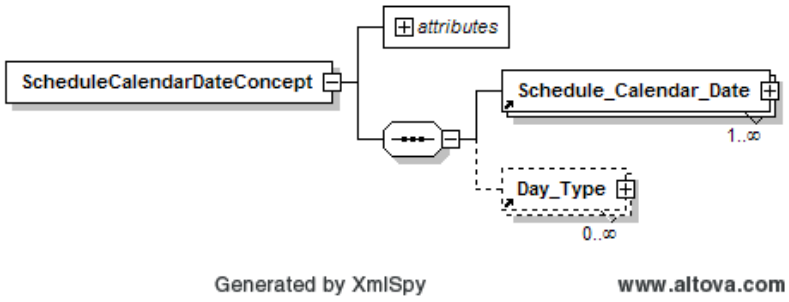

Figure 3-5: Schedule Calendar Date Concept XML Schema

[2] This example does not reflect the current Schedule Calendar Date (SCD) Schema. The DayType element is now included in the SDP XML Schema under the AgencyRegistration branch and is no longer included in the SCD Schema.

Figures 3-6 and 3-7 show the details of each element, Day_Type and Schedule_Calendendar_Date, respectively.

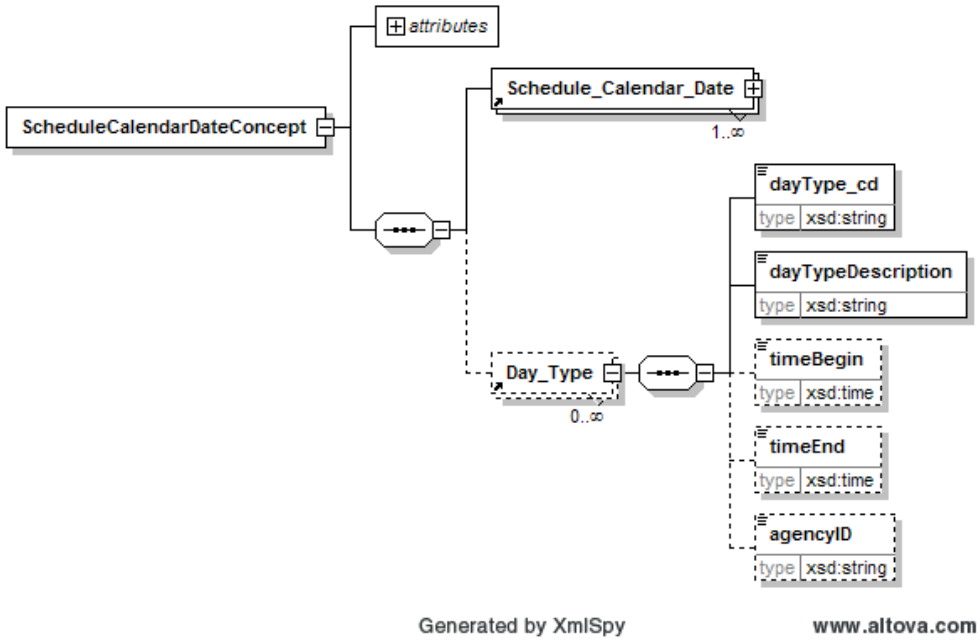

Figure 3-6: Day_Type representation in the Schedule Calendar Date Concept XML Schema

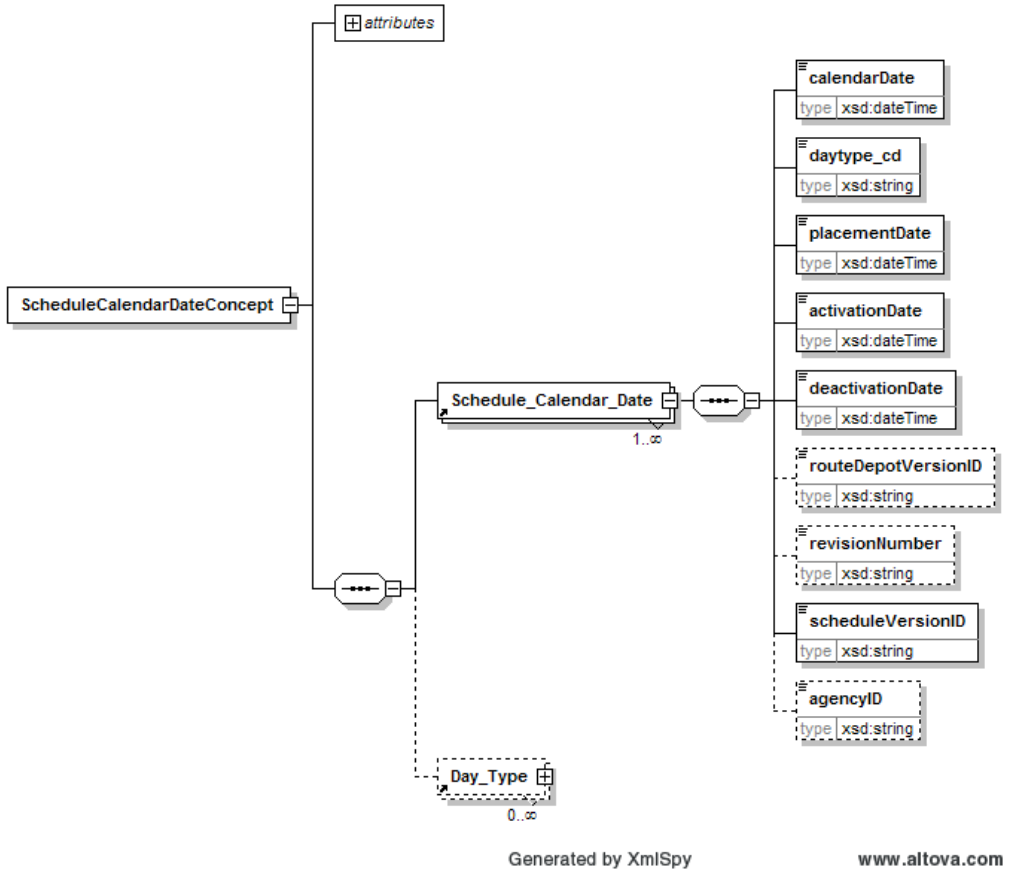

Figure 3-7: Schedule Calendar Date Element in the SCD XML Schema