xsd2pgschema v3 User Guide

Author: Masashi Yokochi, Institute for Protein Research, Osaka University
Updated:December 6, 2018
Target version: v3.0.8

xsd2pgschema is an application suite for processing XML documents in reference to XML Schema. It primarily targets replication of relational database in local environment that (i) reduces network delay and computing resource of the original data publisher and (ii) allows you to optimize response time, which depends on the available resources of yours. Though native XML database provides an one-stop solution for the second purpose. In particular, large data often causes reliability and performance problems. Our tool allows users to select middleware suitable for their own tasks, such as SQL search/update, full-text indexing, and XPath query evaluation. The freedom choice of optimal middleware for specific XML data processing is the next best solution though the following drawbacks are included.

It is noted that the **xsd2pgschema v3** supports XML Schema Definition Language 1.1 (XML Schema 1.1 or XSD 1.1) and XML Path Language 1.0 (XPath 1.0) standards, Unlike the native XML database, XPath 2.0/3.0/3.1, XQuery 1.0/3.0/3.1 standards, and schema-less XML documents are not supported. Instead, our solution enables full-text indexing on XML contents via established open-source search engines (Apache Lucene, or Sphinx Search) and supports wildcards of XSD 1.1 (a.k.a. `xs:any`, and `xs:anyAttribute`), which enable to extend document under the conformity.

This document describes how to install the tool and how to interact with external systems, such PostgreSQL (https://www.postgresql.org), Apache Lucene (http://lucene.apache.org), and Sphinx Search (http://sphinxsearch.com).

License

Requirements

The **xsd2pgschema** is platform-independent and runs on Java 8 (or later) with PostgreSQL 9 (or later). Optionally, Apache Lucene (included in the package), or Sphinx Search can be used as search engine.

Installation

First of all, access https://sourceforge.net/projects/xsd2pgschema/ and download the latest package, `xsd2pgschema-3.x.x.tgz`. Then, uncompress the .tgz file.

```
tar xvzf xsd2pgschema-3.x.x.tgz
```

`xsd2pgschema.jar` file in the package is a JAR file contains main classes, core library and all required libraries. The following commands are typical cases that invoke the main class of the package:

```
java -cp (class_search_path)/xsd2pgschema.jar (main_class_name) (arguments)
```

where `(class_search_path)` is class search path for the JAR file and `(main_class_name)` is the main class name of the package. The default main class is set to `xsd2pgschema`, which converts specified XML Schema to PostgreSQL Data Definition Language (DDL) (see also the section 1.1). `(arguments)` depends on the selected main class, it shares terms in the relevant main classes. See appendix paragraph of each main classes.

Footprint of core library, packaged as `xsd2pgschema-min.jar` file, is small. Users have to gather required libraries (JAR files) as described in `pom.xml`.

Selection of Main Class

The **xsd2pgschema** is originally designed to be a backend tool for construction of a web service of integrated databases which are updated periodically. Individual functions of the tool are available by selection of the main classes as shown. The available functions of the main classes are classified roughly into (1) Replication of PostgreSQL database, (2) Full-text indexing using Apache Lucene or Sphinx Search, (3) File conversion from XML to JSON, (4) XPath 1.0 query evaluation over PostgreSQL, (5) utilities for pre-processing, and (6) Data model server for fast XML Schema analysis (Table. 1). Please jump to the following interesting topics.

Table. 1 Main classes of xsd2pgschema and its functions

Main class	Function	Section
xsd2pgschema	Generate PostgreSQL DDL from XML Schema	1.1
xml2pgsql	PostgreSQL data migration or differential update	1.2
xml2pgcsv/xml2pgtsv	Convert XML to CSV/TSV and data migration (batch)	1.3
csv2pgsql/tsv2pgsql	Import CSV/TSV to PostgreSQL database	1.4
xml2luceneidx	Full-text indexing using Apache Lucene	2.1
luceneidx2dic/ftxt/infix	Generate dictionary from Lucene index	2.2
xml2sphinxds	Prepare Sphinx data source (xmlpipe2) for full-text indexing	2.3
dsmerge4sphinx	Merge Sphinx data sources	2.4
dicmerge4sphinx	Generate dictionary from Sphinx data source	2.5
xsd2jsonschema	Generate JSON Schema from XML Schema	3.1
xml2json	Convert XML documents to JSON documents	3.2
xpathparser	XPath 1.0 parser being aware of XML Schema	4.1
xpath2xml/json/pgsql	XPath 1.0 query evaluation over PostgreSQL	4.2
xmlsplitter	Split large XML file into smaller ones	5.1
xmlvalidator	Parallel XML Schema validation of multiple XML documents	5.2
chksumstat	Report check sum directory status	5.3
pgschemaserv	Data model server for fast XML Schema analysis	6

1. Replication of PostgreSQL database

1.1 Generation of PostgreSQL DDL from XML Schema

`Relavent main class: xsd2pgschema`

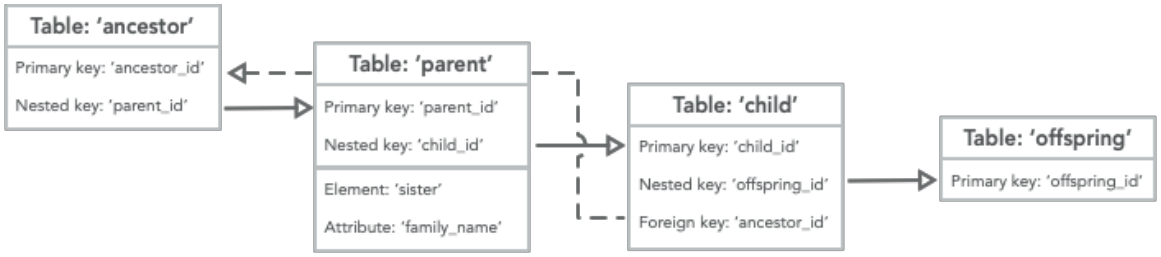
The **xsd2pgschema** main class analyzes XML Schama(ta), generate relational data model internally, and write PostgreSQL DDL file (Appendix. 1). The conversion of hierarchal data model to relational data model is achieved by three **system keys**; **primary key**, **nested key**, **foreign key** (Table. 2). First, all tables have a primary key having concatenated column name pattern, `(table_name)_id`, which serves as destination of the other reference keys. The nested key is like SQL foreign key without a foreign key constraint that points always the primary key of the child table and has concatenated name pattern, `(child_table_name)_id`. The primary key does not always have unique constraint that depends on parent table is list holder, because the single nested key of the list holder (parent) is shared by multiple primary keys of the list member (child). The foreign key has the same concept of SQL term having foreign key constraint. It points the primary key of parent node in ancestor table, and has concatenated name pattern, `(parent_node_name)_id`. The foreign keys are generated when explicit identity-constraint definition, i.e. a pair of `xs:key` and `xs:keyref`, exists or implicit ancestor-child relationship, virtual parent tables in short, is detected in XML Schema analysis. Therefore, foreign key is useful to trace back to parent XML node. Typical hierarchal parent-child relationship is expressed using these system keys are shown for the virtual parent table exists in Fig. 1. The nested key and foreign key have direction from the root node, namely, the nested keys head for leave tables, and the foreign keys head for the parent XML nodes.

Table. 2 System keys to reconstruct hierarchal death model of XML Schema on relational data model

System key	Destination	Column name pattern	Constraint
Primary key		`(table_name)_id`	has a unique constraint except for list member
Nested key	child table's primary key	`(child_table_name)_id`	
Foreign key	parent XML node's primary key in ancestor table	`(parent_node_name)_id`	has a foreign key constraint

where `(table_name)`, `(child_table_name)`, and `(parent_node_name)` represent current table name, child table name, and parent node name, respectively.

Fig. 1 A typical database diagram modeling hierarchal parent-child relationships defined by XML Schema.



XML Schema:

```
<xs:element name="ancestor" type="parent"/>

<xs:complexType name="parent">
  <xs:sequence>
    <xs:element name="child" type="offspring"/>
    <xs:element name="sister" type="xs:string"/>
 </xs:sequence>
 <xs:attribute name="family_name" type="xs:string"/>
</xs:complexType>
```

XML Instance:

```
<ancestor family_name="…">
 <child>
  …
 </child>
 <sister>…</sister>
</ancestor>
```

Optionally, the relational data model can be supported by introduction of three **user keys** to facilitating SQL search or update, preserving document order, and enabling absolute XPath addressing. They are called **document key**, **serial key**, and **xpath key**, respectively (Table. 4). The document key has text data filled with a value quoted from source XML document file name, which represents unique ID for each unit of data, such as entry ID or session ID. The document key is appended to all tables by default because the document ID often becomes primary SQL query's condition or SQL search result. The serial key holds ordinal numbers that reveal document order in list. You may need the serial key to execute document order dependent XPath queries. Though, PostgreSQL does not assure document order in data migration using `COPY` command, the document order will be practically preserved when you use the **xml2pgsql** main class for the data migration. The last xpath key provides a way to select rows from absolute XPath addressing. To reduce database size, the xpath key only holds hash codes representing the absolute XPath of current node. Please refer to Fig. 2 for the encoding scheme when document ID is empty.

Table. 3 User keys to facilitate SQL search or update

User key	Role	Column name	Data type	Defalt
Document key	specify source XML document	document_id	text	yes
Serial key	document order preservation	serial_id	ordinal number	no
XPath key	absolute XPath addressing	xpath_id	hash code	no

As shown Table 2 and 3, there are reserved column name for system keys and user keys. In case that schema component name of XML Schema matches the reserved names, the tool will rename the schema component name by inserting an under score character, "_", at a head of the schema component name. When name collision between tables occurs, the tool will merge tables into one. However, it rarely occurs because XML namespace is supported. In general, name convention of XML Schema is kept with the best effort. The tool also keeps several reserved column names for special data types, `xs:simpleContent`, `xs:any`, and `xs:anyAttribute`. (Table. 4)

Table. 4 Mapping column name for special data types

Data type	Role	Column name	Value type
xs:simpleContent	Simple content	content	text
xs:any	wild card for any element	any_element	xml
xs:anyAttribute	wild card for any attribute	any_attribute	xml

In case the document key become redundant, you can take over document key's role to the existing element, attribute, or simple content that is called in-place document key. If you don't want to use relational data model extension using the system keys, you can cancel the relational data model extension. But, no support for data update is available without the document key.

Finally, the generated DDL file is applied to PostgreSQL database by the following `psql` command:

```
createdb -U (db_user) (db_name) # run the first time only
psql -d (db_name) -U (db_user) -f (ddl_file)
```

where `(db_name)` is database name, and `(db_user)` is database user name. You must have permission to create database, of course. `(ddl_file)` represents the generated PostgreSQL DDL file.

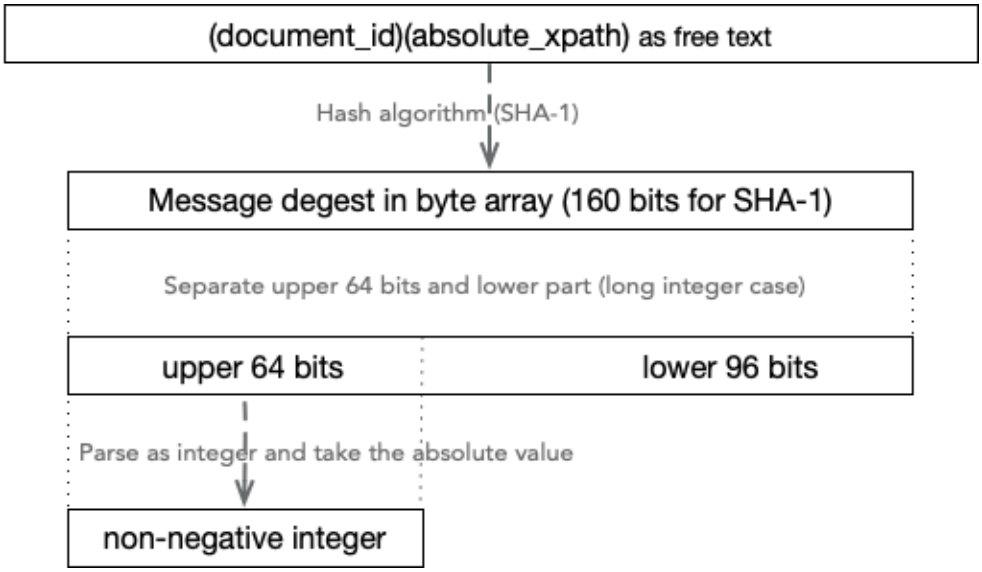
1.2 PostgreSQL data migration or differential update

```
Relavent main class: xml2pgsql
```

The **xml2pgsql** main class parses multiple XML documents using document object model (DOM), then imports to PostgreSQL database defined by the generated PostgreSQL DDL as shown in the previous section. You must select the same XML Schema(ta) and set all relevant arguments used in the **xsd2pgschema** main class (Appendix. 2).
The document ID will be set by quoting XML file names ignoring its file extension such as `.xml`, `.xml.gz`, and `.xml.zip`. Please specify the file extension using **--xml-file-**

ext FILE_EXTENSION argument, there is no need to uncompress the XML files. If you properly select the XML file extension, the tool finds all XML files having the same file extension in directories specified by **--xml XML_FILE_OR_DIRECTORY** arguments and uncompress internally. The document ID can be edited by combination of the following arguments: **--xml-file-prerix-digest DIGESTIBLE_PREFIX**, **--xml-file-ext-digest DIGESTIBLE_EXTENSION**, **--lower-case-doc-key**, and **—upper-case-doc-key**. Data type of the system keys are defined by both **--hash-by ALGORITHM** and **--hash-size BIT_SIZE** arguments. By default, non-negative 64 bits integer, information amount equivalent to 63 bits, is selected. Detailed encoding scheme of the system keys is shown in Fig. 2.

Fig. 2 A typical encoding scheme for generation of the system keys.



The value of system key depends on document ID and absolute location of XML node written in XPath expression. Though, uniqueness of key is not assured by the bit truncation after the message digest, hash collision rarely occurs because message digest will show drastic change with tiny changes and source key is limited on particular part of documents. Moreover, PostgreSQL can detect violation of unique constraint while data migration. For the case, the tool also allows users to select the native length of byte array as data type of the system keys. The use of 64 bits integer usually is enough and practical choice for performance and storage size because the standard cryptographic hash calculations tend to be accelerated by specific circuits implemented in currently available CPUs.

1.3 Convert XML to CSV/TSV and data migration

```
Relavent main class: xml2pgcsv, xml2pgtsv
```

The **xml2pgcsv** and **xml2pgtsv** are sister main classes, which convert XML documents to CSV/TSV files for each relations, then execute PostgreSQL's COPY command if database is specified. This is the fastest method of data migration. The **--sync CHECK_SUM_DIRECTORY** argument, see Appendix. 3, is implemented to enable differential update for the next time by the **xml2pgsql** main class. It is noted again that PostgreSQL does not assure document order in the batch data migration and you must select the same XML Schema(ta) and set all relevant arguments used in the **xsd2pgschema** main class.

1.4 Import CSV/TSV and data migration

```
Relavent main class: csv2pgsql, tsv2pgsql
```

The **csv2pgsql** and **tsv2pgsql** are sister main classes, which import CSV/TSV files in a working directory to PostgreSQL database. These main classes should be invoked for CSV/TSV files generated by either **xml2pgcsv** or **xml2pgtsv** main class, respectively. There is no substantial difference as data migration in separating into two steps, generates CSV/TSV files, then imports to PostgreSQL database.
These main classes is prepared for debugging on the system keys, we can check human readable system key values in the CSV files generated by the **xml2pgcsv** main class with **--hash-size debug** argument in Appendix. 3. Then, we can test the PostgreSQL's COPY command step by step.

2. Full-text indexing using Apache Lucene or Sphinx Search

2.1 Full-text index using Apache Lucene

```
Relavent main class: xml2luceneidx
```

The **xml2luceneidx** main class parses multiple XML documents and generate full-text indexing using Apache Lucene. Apache Lucene provides full functions as search engine including full-text indexing, query parser, auto-suggestion, highlighting, and so on. In order to build a web service from XML documents, a combination of a search engine and a relational database becomes essential software stack. The main class supports full-text indexing of whole XML documents, differential update of index, field selection for indexing by means of SQL like designation, `(table_name).(column_name)`, attribute selection that defines retrievable value stored in index, content type dependent attribute selection, setting minimum words required for indexing, and sharding (Appendix. 5). Under normal settings, the tool performs full-text indexing on an index field named “content” for all XML content. You can append any index attributes to retrieve associated values to hit documents like simple database. For information on how to use the generated Lucene index, please refer to the official document.

2.2 Generate dictionary from Lucene index

```
Relavent main class: luceneidx2dic, luceneidx2ftxt, luceneidx2in-
fix
```

The **luceneidx2dic** , **luceneidx2ftxt**, **luceneidx2infix** main classes are utility dedicated for providing auto-suggestion function on supposed web service. They open Lucene index and generate dictionary for specified fields. For information on how to implement the dictionary to your service, please refer to the official document.

2.3 Prepare Sphinx data source (xmlpipe2) for full-text indexing

```
Relavent main class: xml2sphinxds
```

The **xml2pshinxds** main class parses multiple XML documents and generate Sphinx data source in xmlpipe2 format. Sphinx is an alternative open-source search engine implemented by in-memory technologies so that it often excels in search performance if there is plenty of memory. It has own query language, SphinxQL, acting as a drop-in extension on MySQL query interface. The xmlpipe2 format is one of data source for Sphinx. In actual, The full-text indexing is actually applied on the generated data source by Sphinx’s command, `indexer`. As with the Lucune’s full-text indexing, the main class supports differential update of the data source, field selection for indexing by means of SQL like designation, `(table_name).(column_name)`, attribute selection that defines retrievable value stored in index, content type dependent attribute selection, setting minimum words required for indexing, and sharding (Appendix. 7). “content” is the field index for all XML content. It is noted that there is a little difference

in attribute designation in SphinxQL, please use member operator "__" in SphinxQL to avoid confusion with SQL's ".". For example, the attribute in SphinxQL should be expressed by `(table_name)__(column_name)`. The documents in the xmlpipe2 format are expressed in a tag "<sphinx:document id='...'>" where the attribute "id" having a unique 64 bits integer, which is filled with the code corresponding the document key values using the same encoding scheme shown in Fig. 2. Please refer to the official document for Sphinx and SphinxQL.

2.4 Merge Sphinx data sources

```
Relavent main class: dsmerge4sphinx
```

The **dsmerge4sphinx** main class merges multiple data sources into one for the case the sharding is unnecessary. It is noted that the sharding is required for big data source because maximum memory allocation for a data source is limited. See also Appendix 7 and 8.

2.5 Generate dictionary from Sphinx data sources

```
Relavent main class: dicmerge4sphinx
```

The **dicmerge4sphinx** main class generates dictionary from Sphinx data source. It is counterpart of lucene2dic/ftxt/infix main classes. Generation of a dictionary from index is not implemented as a Sphinx function so far. For auto-suggestion function, the main class generates the dictionary by converting keywords in the source index to trigrams.

3. File conversion from XML to JSON

3.1 Generate JSON Schema from XML Schema

```
Relavent main class: xsd2jsonschema
```

The **xsd2jsonschema** main class maps XML schema components to JSON schema components. JSON has advantages being light-weight as data container and having flexibility for schema-less document in comparison with XML. The flexibility often leads negative consequence as protocol of web service. JSON Schema (https://json-schema.org) provides one of solutions that define document and web service. It is noted that JSON Schema is not standardized by the IETF, Internet Engineering Task Force as of 2018. We focus on document definition using JSON Schema. There is no straight mapping scheme from XML Schema to JSON Schema that depends on how to deal XML's simple content and JSON's array in the conversion. The tool provides the following three JSON Schema mappings: object-oriented, column-oriented, and relational-oriented schema mapping. First, the object-oriented schema mapping is an intuitive translation that holds the same XML data structure in JSON. Second, the column-oriented schema mapping utilizes JSON array for storage values of list members, which can reduce redundant XML tags in a list effectively and selected this mapping by default. Finally, the relational-oriented schema mapping also utilizes JSON array for all values of any relations just like dumping data of relational database so that the definition of JSON objects is minimum. The system keys and user keys are ignored in the JSON Schema mapping and JSON document conversion. In either case, the simple content of the XML document creates a JSON object named "content". Please see Appendix. 10 for other arguments. It is possible to validate the generated JSON Schema against JSON Schema core specifications using online JSON Schema validation service. Because the tool parses XML Schema and then create an instance of relational data model, Re-usable schema components in XML Schema such as `<xs:complexType>` will not converted to counterparts directory of the JSON Schema. Therefore, the obtained JSON Schema may have redundant JSON object definitions. Instead of the redundancy, we can choose the different JSON Schema mapping strategies for arbitrary complex XML Schema(ta).

3.2 Convert XML documents to JSON documents

```
Relavent main class: xml2json
```

The **xml2json** main class converts selected XML documents to JSON documents. Unlike other main classes, the main class does not support differential update because generated JSON document depends on not only source XML documents but also the selected arguments shown in Appendix. 11. It would be better to validate the obtained JSON documents against the JSON Schema converted from the XML Schema.

4. XPath 1.0 query evaluation over PostgreSQL

4.1 XPath 1.0 parser being aware of XML Schema

```
Relavent main class: xpathparser
```

The **xpathparser** main class parses XPath 1.0 query based on ANTLR v4 (https://www.antlr.org) and reports abstract syntax tree of the query and validity against the relational data model. (Appendix. 12)

4.2 XPath 1.0 query evaluation over PostgreSQL

```
Relavent main class: xpath2xml, xpath2json, xpath2pgsql
```

The **xpath2xml**, **xpath2json**, **xpath2pgsql** main classes parses XPath 1.0 query and evaluate the query on the PostgreSQL database generated by the data migration, then output results as XML, JSON, or CSV/TSV formats, respectively. (Appendix. 13, 14, 15) At first, the tool performs query translation from XPath to SQL, then, executes the SQL query. All child nodes in the results are traced until they reach terminus nodes. In most cases, the final results forms a fragmented document, which contains multiple root nodes. There is no need to set target namespace URI and prefix, they are defined by the source XML Schema. It is possible append document ID under the root node of the result. It is easiest ways to get the source documents. Some queries depending on document order requires that the serial keys must be prepared when the data migration. Users can run multiple XPath evaluations and the results are stored in different files for each XPath query. For optimal performance as a web service, instance objects in the main class should be reused because Java process's startup time and XML Schema analysis time are not ignorable. Please utilize a data model server, called **PgSchema server** (**pgschemaserv** main class), used for acceleration of the XML Schema analysis (see the 6th section). Please find example scripts to control the PgSchema server; `start_pgschema_serv.sh`, `stop_pgschema_serv.sh`, and `status_pgschema_serv.sh`. Of course, optimizing PostgreSQL using index also improves real performance, too. There is a function to support creation of PostgreSQL's indexes (see Appendix 2 and 3), but it is lefts to user's discretions.

5. Utilities for pre-processing

5.1 Split large XML file to smaller ones

```
Relavent main class: xmlsplitter
```

The **xmlsplitter** main class splits a large XML file into smaller ones. The large XML file could not be parsed by DOM since memory limitation. Users can arbitrary decide the location of document keys using XPath expression, after that values of the document key become split XML file names (Appendix. 16). It is not like a simple chopper, but split XML documents are valid against the XML Schema, too. Please find an example script, `split_uniorotkb.sh`, used for splitting a large XML file.

5.2 Parallel XML Schema validation on multiple XML documents

```
Relavent main class: xmlvalidator
```

The **xmlvalidator** main class is a helper tool for XML Schema validation based on Apache Xerces (http://xerces.apache.org). It is a heavy task to validate a large number of XML documents and use complex XML Schema for validation. The main class supports parallel XML Schema validation and differential update, You can use available computer resources to speed up tasks and reduce the tasks of previously validated XML documents. Optionally, it is possible to remove invalid XML documents (Appendix. 17).

5.3 Report check sum directory status

```
Relavent main class: chksumstat
```

The **chksumstat** main class compares the checksum of previous processing and reports which XML documents is created, updated, or deleted before actual processing begins (Appendix. 18).

6. Data model server for faster XML Schema analysis

```
Relavent main class: pgschemaserv
```

The **pgschemaserv** main class is a server implementation that provides a previously generated data model to the client main class. Analysis of complex XML Schema(ta) and succeeding data model generation cost a lot. Because all main classes that process multiple XML documents are parallelized, the same data model is generated for each thread. Furthermore, XPath query evaluation must be fast. The **PgSchema server** is a dedicated server to improve performance of the xsd2pgschema tools. Please find the server control scripts in example directory (see also the 4th section), then you can test performance when the server is alive (Appendix. 18). The PgSchema server uses the 5430 port, and the client main classes are enabled to access the server by default.

Appendixes

The **xsd2pgschema** is an application suite application for processing XML documents based on XML Schema. Individual function of the tool can be accessed through the following main classes and their arguments.

Appendix. 1 Arguments of the **xsd2pgschema** main class

```
xsd2pgschema: XML Schema -> PostgreSQL DDL conversion
Usage:  --xsd SCHEMA_LOCATION --ddl DDL_FILE (default=stdout)
        --no-rel (turn off relational model extension)
        --no-wild-card (turn off wild card extension)
        --doc-key (append document_id column in all relations, default
with relational model extension)
        --no-doc-key (remove document_id column from all relations,
effective only with relational model extension)
        --ser-key (append serial_id column in child relation of list
holder)
        --xpath-key (append xpath_id column in all relations)
        --no-key (turn off constraint of primary key/foreign key/
unique)
Option: --case-insensitive (all table and column names are lowercase)
        --pg-public-schema (utilize "public" schema, default)
        --pg-named-schema (enable explicit named schema)
        --pg-map-big-integer (map xs:integer to BigInteger according
to the W3C rules)
        --pg-map-long-integer (map xs:integer to signed long 64 bits)
        --pg-map-integer (map xs:integer to signed int 32 bits, de-
fault)
        --pg-map-big-decimal (map xs:decimal to BigDecimal according
to the W3C rules, default)
        --pg-map-double-decimal (map xs:decimal to double precision 64
bits)
        --pg-map-float-decimal (map xs:decimal to single precision 32
bits)
        --field-annotation (retrieve field annotation)
        --no-field-annotation (do not retrieve field annotation, de-
fault)
        --max-uniq-touple-size MAX_UNIQ_TUPLE_SIZE (maximum tuple size
of unique constraint derived from xs:key, ignore the limit if non-pos-
itive value, default=1)
        --no-cache-xsd (retrieve XML Schemata without caching)
        --hash-by ASSUMED_ALGORITHM [MD2 | MD5 | SHA-1 (default) |
SHA-224 | SHA-256 | SHA-384 | SHA-512]
        --hash-size BIT_SIZE [int (32bit) | long (64bit, default) |
native (default bit of algorithm) | debug (string)]
        --ser-size BIT_SIZE [short (16bit); | int (32bit, default)]
        --doc-key-name DOC_KEY_NAME (default="document_id")
        --ser-key-name SER_KEY_NAME (default="serial_id")
        --xpath-key-name XPATH_KEY_NAME (default="xpath_id")
        --discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
        --inplace-doc-key-name INPLACE_DOCUMENT_KEY_NAME
        --doc-key-if-no-inplace (append document key if no in-place
document key, select --no-doc-key options by default)
        --show-orphan-table (map orphan tables)
```

Appendix 2. Arguments of the **xml2pgsql** main class

```
xml2pgsql: XML -> PostgreSQL data migration
Usage:  --xsd SCHEMA_LOCATION --xml XML_FILE_OR_DIRECTORY --db-name
DATABASE --db-user USER --db-pass PASSWORD (default="")
        --db-host PG_HOST_NAME (default="localhost")
        --db-port PG_PORT_NUMBER (default=5432)
        --test-ddl (perform consistency test on PostgreSQL DDL)
        --min-rows-for-index MIN_ROWS_FOR_INDEX (default=2048)
        --create-doc-key-index (create PostgreSQL index on document
key if not exists, enable if --sync option is selected)
        --no-create-doc-key-index (do not create PostgreSQL index on
document key, default if no --sync option)
        --drop-doc-key-index (drop PostgreSQL index on document key if
exists)
        --create-attr-index (create PostgreSQL index on attribute if
not exists, default)
        --no-create-attr-index (do not create PostgreSQL index on at-
tribute)
        --drop-attr-index (drop PostgreSQL index on attribute if ex-
ists)
        --max-attr-cols-for-index MAX_ATTR_COLS_FOR_INDEX (default=1)
        --create-elem-index (create PostgreSQL index on element if not
exists)
        --no-create-elem-index (do not create PostgreSQL index on ele-
ment, default)
        --drop-elem-index (drop PostgreSQL index on element if exists)
        --max-elem-cols-for-index MAX_ELEM_COLS_FOR_INDEX (default=1)
        --create-simple-cont-index (create PostgreSQL index on simple
content if not exists, default)
        --no-create-simple-cont-index (do not create PostgreSQL index
on simple content)
        --drop-simple-cont-index (drop PostgreSQL index on simple con-
tent if exists)
        --max-fks-for-simple-cont-index MAX_FKS_FOR_SIMPLE_CONT_INDEX
(default=0)
        --update (insert if not exists, and update if required, de-
fault)
        --sync CHECK_SUM_DIRECTORY (insert if not exists, update if
required, and delete rows if XML not exists, select --create-doc-key-
index option by default)
        --sync-weak (insert if not exists, no update even if exists,
no deletion, select --create-doc-key-index option by default)
        --sync-rescue (diagnostic synchronization, set all constraints
deferred)
        --no-rel (turn off relational model extension)
        --no-wild-card (turn off wild card extension)
        --doc-key (append document_id column in all relations, default
with relational model extension)
```

--no-doc-key (remove document_id column from all relations, effective only with relational model extension)
--ser-key (append serial_id column in child relation of list holder)
--xpath-key (append xpath_id column in all relations)
--no-key (turn off constraint of primary key/foreign key/ unique)
--validate (turn on XML Schema validation)
--no-validate (turn off XML Schema validation, default)
--well-formed (validate only whether document is well-formed)
--xml-file-ext FILE_EXTENSION [xml (default) | gz (indicates xml.gz suffix) | zip (indicates xml.zip suffix)]
Option: **--case-insensitive** (all table and column names are lowercase)
--pg-public-schema (utilize "public" schema, default)
--pg-named-schema (enable explicit named schema)
--pg-map-big-integer (map xs:integer to BigInteger according to the W3C rules)
--pg-map-long-integer (map xs:integer to signed long 64 bits)
--pg-map-integer (map xs:integer to signed int 32 bits, default)
--pg-map-big-decimal (map xs:decimal to BigDecimal according to the W3C rules, default)
--pg-map-double-decimal (map xs:decimal to double precision 64 bits)
--pg-map-float-decimal (map xs:decimal to single precision 32 bits)
--no-cache-xsd (retrieve XML Schemata without caching)
--checksum-by ALGORITHM [MD2 | MD5 (default) | SHA-1 | SHA-224 | SHA-256 | SHA-384 | SHA-512]
--hash-by ALGORITHM [MD2 | MD5 | SHA-1 (default) | SHA-224 | SHA-256 | SHA-384 | SHA-512]
--hash-size BIT_SIZE [int (32bit) | long (64bit, default) | native (default bit of algorithm) | debug (string)]
--ser-size BIT_SIZE [short (16bit) | int (32bit, default)]
--xml-file-prerix-digest DIGESTIBLE_PREFIX (default="")
--xml-file-ext-digest DIGESTIBLE_EXTENSION (default=".")
--lower-case-doc-key (lower case document key)
--upper-case-doc-key (upper case document key)
--fill-default-value (fill @default value in case of empty)
--filt-in table_name.column_name
--filt-out table_name.column_name:regex_pattern(|regex_pattern...)
--fill-this table_name.column_name:filling_text
--doc-key-name DOC_KEY_NAME (default="document_id")
--ser-key-name SER_KEY_NAME (default="serial_id")
--xpath-key-name XPATH_KEY_NAME (default="xpath_id")
--discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
--inplace-doc-key-name INPLACE_DOCUMENT_KEY_NAME

--doc-key-if-no-inplace (append document key if no in-place document key, select --no-doc-key options by default)
--no-pgschema-serv (not utilize PgSchema server)
--pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="local-host")
--pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
--max-thrds MAX_THRDS (default is number of available proces-sors)

Appendix 3. Arguments of the **xml2pgcsv** (or **xml2pgtsv**) main class

```
xml2pgcsv: XML -> CSV conversion and PostgreSQL data migration
Usage:  --xsd SCHEMA_LOCATION --xml XML_FILE_OR_DIRECTORY --work-dir
DIRECTORY (default="pg_work")
        --no-rel (turn off relational model extension)
        --no-wild-card (turn off wild card extension)
        --doc-key (append document_id column in all relations, default
with relational model extension)
        --no-doc-key (remove document_id column from all relations,
effective only with relational model extension)
        --ser-key (append serial_id column in child relation of list
holder)
        --xpath-key (append xpath_id column in all relations)
        --no-key (turn off constraint of primary key/foreign key/
unique)
        --validate (turn on XML Schema validation)
        --no-validate (turn off XML Schema validation, default)
        --well-formed (validate only whether document is well-formed)
        --xml-file-ext FILE_EXTENSION [xml (default) | gz (indicates
xml.gz suffix) | zip (indicates xml.zip suffix)]
Option: --db-name DATABASE --db-user USER --db-pass PASSWORD
(default="")
        --db-host PG_HOST_NAME (default="localhost")
        --db-port PG_PORT_NUMBER (default=5432)
        --test-ddl (perform consistency test on PostgreSQL DDL)
        --min-rows-for-index MIN_ROWS_FOR_INDEX (default=2048)
        --create-doc-key-index (create PostgreSQL index on document
key if not exists, enable if --sync option is selected)
        --no-create-doc-key-index (do not create PostgreSQL index on
document key, default if no --sync option)
        --drop-doc-key-index (drop PostgreSQL index on document key if
exists)
        --create-attr-index (create PostgreSQL index on attribute if
not exists, default)
        --no-create-attr-index (do not create PostgreSQL index on at-
tribute)
        --drop-attr-index (drop PostgreSQL index on attribute if ex-
ists)
        --max-attr-cols-for-index MAX_ATTR_COLS_FOR_INDEX (default=1)
        --create-elem-index (create PostgreSQL index on element if not
exists)
        --no-create-elem-index (do not create PostgreSQL index on ele-
ment, default)
        --drop-elem-index (drop PostgreSQL index on element if exists)
        --max-elem-cols-for-index MAX_ELEM_COLS_FOR_INDEX (default=1)
        --create-simple-cont-index (create PostgreSQL index on simple
content if not exists, default)
```

--no-create-simple-cont-index (do not create PostgreSQL index on simple content)
--drop-simple-cont-index (drop PostgreSQL index on simple content if exists)
--max-fks-for-simple-cont-index MAX_FKS_FOR_SIMPLE_CONT_INDEX (default=0)
--case-insensitive (all table and column names are lowercase)
--pg-public-schema (utilize "public" schema, default)
--pg-named-schema (enable explicit named schema)
--pg-map-big-integer (map xs:integer to BigInteger according to the W3C rules)
--pg-map-long-integer (map xs:integer to signed long 64 bits)
--pg-map-integer (map xs:integer to signed int 32 bits, default)
--pg-map-big-decimal (map xs:decimal to BigDecimal according to the W3C rules, default)
--pg-map-double-decimal (map xs:decimal to double precision 64 bits)
--pg-map-float-decimal (map xs:decimal to single precision 32 bits)
--pg-tab-delimiter (use tab separated file)
--no-cache-xsd (retrieve XML Schemata without caching)
--sync CHECK_SUM_DIRECTORY (generate check sum files for differential update, select --create-doc-key-index option by default)
--checksum-by ALGORITHM [MD2 | MD5 (default) | SHA-1 | SHA-224 | SHA-256 | SHA-384 | SHA-512]
--hash-by ALGORITHM [MD2 | MD5 | SHA-1 (default) | SHA-224 | SHA-256 | SHA-384 | SHA-512]
--hash-size BIT_SIZE [int (32bit) | long (64bit, default) | native (default bit of algorithm) | debug (string)]
--ser-size BIT_SIZE [short (16bit) | int (32bit, default)]
--xml-file-prerix-digest DIGESTIBLE_PREFIX (default="")
--xml-file-ext-digest DIGESTIBLE_EXTENSION (default=".")
--lower-case-doc-key (lower case document key)
--upper-case-doc-key (upper case document key)
--fill-default-value (fill @default value in case of empty)
--filt-in table_name.column_name
--filt-out table_name.column_name:regex_pattern(|regex_pattern...)
--fill-this table_name.column_name:filling_text
--doc-key-name DOC_KEY_NAME (default="document_id")
--ser-key-name SER_KEY_NAME (default="serial_id")
--xpath-key-name XPATH_KEY_NAME (default="xpath_id")
--discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
--inplace-doc-key-name INPLACE_DOCUMENT_KEY_NAME
--doc-key-if-no-inplace (append document key if no in-place document key, select --no-doc-key options by default)
--no-pgschema-serv (not utilize PgSchema server)

--pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="local-host")
--pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
--max-thrds MAX_THRDS (default is number of available proces-sors)

Appendix 4. Arguments of the **csv2pgsql** (or **tsv2pgsql**) main class

```
csv2pgsql: CSV -> PostgreSQL data migration
Usage:  --xsd SCHEMA_LOCATION --work-dir DIRECTORY (default="pg_work")
--db-name DATABASE --db-user USER --db-pass PASSWORD (default="")
        --db-host PG_HOST_NAME (default="localhost")
        --db-port PG_PORT_NUMBER (default=5432)
        --test-ddl (perform consistency test on PostgreSQL DDL)
        --min-rows-for-index MIN_ROWS_FOR_INDEX (default=2048)
        --create-doc-key-index (create PostgreSQL index on document
key if not exists, enable if --sync option is selected)
        --no-create-doc-key-index (do not create PostgreSQL index on
document key, default if no --sync option)
        --drop-doc-key-index (drop PostgreSQL index on document key if
exists)
        --create-attr-index (create PostgreSQL index on attribute if
not exists, default)
        --no-create-attr-index (do not create PostgreSQL index on at-
tribute)
        --drop-attr-index (drop PostgreSQL index on attribute if ex-
ists)
        --max-attr-cols-for-index MAX_ATTR_COLS_FOR_INDEX (default=1)
        --create-elem-index (create PostgreSQL index on element if not
exists)
        --no-create-elem-index (do not create PostgreSQL index on ele-
ment, default)
        --drop-elem-index (drop PostgreSQL index on element if exists)
        --max-elem-cols-for-index MAX_ELEM_COLS_FOR_INDEX (default=1)
        --create-simple-cont-index (create PostgreSQL index on simple
content if not exists, default)
        --no-create-simple-cont-index (do not create PostgreSQL index
on simple content)
        --drop-simple-cont-index (drop PostgreSQL index on simple con-
tent if exists)
        --max-fks-for-simple-cont-index MAX_FKS_FOR_SIMPLE_CONT_INDEX
(default=0)
        --no-rel (turn off relational model extension)
        --no-wild-card (turn off wild card extension)
        --doc-key (append document_id column in all relations, default
with relational model extension)
        --no-doc-key (remove document_id column from all relations,
effective only with relational model extension)
        --ser-key (append serial_id column in child relation of list
holder)
        --xpath-key (append xpath_id column in all relations)
Option: --case-insensitive (all table and column names are lowercase)
        --pg-public-schema (utilize "public" schema, default)
        --pg-named-schema (enable explicit named schema)
```

--pg-map-big-integer (map xs:integer to BigInteger according to the W3C rules)
--pg-map-long-integer (map xs:integer to signed long 64 bits)
--pg-map-integer (map xs:integer to signed int 32 bits, default)
--pg-map-big-decimal (map xs:decimal to BigDecimal according to the W3C rules, default)
--pg-map-double-decimal (map xs:decimal to double precision 64 bits)
--pg-map-float-decimal (map xs:decimal to single precision 32 bits)
--pg-tab-delimiter (use tab separated file)
--no-cache-xsd (retrieve XML Schemata without caching)
--doc-key-name DOC_KEY_NAME (default="document_id")
--ser-key-name SER_KEY_NAME (default="serial_id")
--xpath-key-name XPATH_KEY_NAME (default="xpath_id")
--discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
--inplace-doc-key-name INPLACE_DOCUMENT_KEY_NAME
--doc-key-if-no-inplace (append document key if no in-place document key, select --no-doc-key options by default)
--no-pgschema-serv (not utilize PgSchema server)
--pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="localhost")
--pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)

Appendix 5. Arguments of the **xml2luceneidx** main class

```
xm2luceneidx: XML -> Lucene full-text indexing
Usage:  --xsd SCHEMA_LOCATION --xml XML_FILE_OR_DIRECTORY --idx-dir
DIRECTORY (default="lucene_index")
        --update (insert if not exists, and update if required, de-
fault)
        --sync CHECK_SUM_DIRECTORY (insert if not exists, update if
required, and delete rows if XML not exists)
        --sync-weak (insert if not exists, no update even if exists,
no deletion)
        --rel (turn on relational model extension)
        --no-rel (turn off relational model extension, default)
        --no-wild-card (turn off wild card extension)
        --validate (turn on XML Schema validation)
        --no-validate (turn off XML Schema validation, default)
        --well-formed (validate only whether document is well-formed)
        --xml-file-ext FILE_EXTENSION [xml (default) | gz (indicates
xml.gz suffix) | zip (indicates xml.zip suffix)]
        --shard-size SHARD_SIZE (default=1)
        --min-word-len MIN_WORD_LENGTH (default is 1)
        --numeric-idx (allow to store numeric values in index)
Option: --attr  table_name.column_name
        --field table_name.column_name
        --field-all (index all fields, default)
        --attr-all (all attributes's values are stored as attribute )
        --attr-string (all string values are stored as attribute)
        --attr-integer (all integer values are stored as attribute)
        --attr-float (all float values are stored as attribute)
        --attr-date (all date values are stored as attribute)
        --attr-time (all time values are stored as attribute)
        --no-cache-xsd (retrieve XML Schemata without caching)
        --hash-by ALGORITHM [MD2 | MD5 | SHA-1 (default) | SHA-224 |
SHA-256 | SHA-384 | SHA-512]
        --hash-size BIT_SIZE [int (32bit) | long (64bit, default) |
native (default bit of algorithm) | debug (string)]
        --xml-file-prerix-digest DIGESTIBLE_PREFIX (default="")
        --xml-file-ext-digest DIGESTIBLE_EXTENSION (default=".")
        --lower-case-doc-key (lower case document key)
        --upper-case-doc-key (upper case document key)
        --fill-default-value (fill @default value in case of empty)
        --filt-in   table_name.column_name
        --filt-out  table_name.column_name:regex_pattern(|regex_pat-
tern...)
        --fill-this table_name.column_name:filling_text
        --discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
        --no-pgschema-serv (not utilize PgSchema server)
        --pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="local-
host")
```

--pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
--max-thrds MAX_THRDS (default is number of available processors)

Appendix 6.1. Arguments of the **luceneidx2dic** (or **luceneidx2ftxt**) main class

```
luceneidx2dic: Lucene index -> Lucene dictionary
Usage:  --idx-dir DIRECTORY (default="lucene_index") --dic-dir DIREC-
TORY (default="lucene_dic") --dic DIC_FILE (default="dictionary")
Option: --field FIELD_NAME (default="content")
        --freq FREQ_THRESHOLD (default=10)
```

Appendix 6.2. Arguments of the **luceneidx2ftxt** main class

```
luceneidx2infix: Lucene index -> Lucene analyzed infix suggester
Usage:  --idx-dir DIRECTORY (default="lucene_index") --infix-dir DI-
RECTORY (default="lucene_infix")
Option: --field FIELD_NAME (default="content")
        --freq FREQ_THRESHOLD (default=10)
```

Appendix 7. Arguments of the **xml2sphinxds** main class

```
xml2sphinxds: XML -> Sphinx data source (xmlpipe2) conversion for
full-text indexing
Usage:  --xsd SCHEMA_LOCATION --xml XML_FILE_OR_DIRECTORY --ds-dir DI-
RECTORY (default="sphinx_xmlpipe2")
        --update (insert if not exists, and update if required, de-
fault)
        --sync CHECK_SUM_DIRECTORY (insert if not exists, update if
required, and delete rows if XML not exists)
        --sync-weak (insert if not exists, no update even if exists,
no deletion)
        --no-wild-card (turn off wild card extension)
        --validate (turn off XML Schema validation)
        --no-validate (turn off XML Schema validation, default)
        --well-formed (validate only whether document is well-formed)
        --xml-file-ext FILE_EXTENSION [xml (default) | gz (indicates
xml.gz suffix) | zip (indicates xml.zip suffix)]
        --shard-size SHARD_SIZE (default=1)
        --min-word-len MIN_WORD_LENGTH (default is 1)
        --max-field-len MAX_FIELD_LENGTH (default is 2M)
Option: --ds-name DS_NAME (default name is determined by quoting XSD
file name)
        --attr  table_name.column_name
        --field table_name.column_name
        --mva   table_name.column_name (multi-valued attribute)
        --field-all (index all fields, default)
        --attr-all (all attributes's values are stored as attribute )
        --attr-string (all string values are stored as attribute)
        --attr-integer (all integer values are stored as attribute)
        --attr-float (all float values are stored as attribute)
        --attr-date (all date values are stored as attribute)
        --attr-time (all time values are stored as attribute)
        --no-cache-xsd (retrieve XML Schemata without caching)
        --hash-by ALGORITHM [MD2 | MD5 | SHA-1 (default) | SHA-224 |
SHA-256 | SHA-384 | SHA-512]
        --hash-size BIT_SIZE [int | long (default) | native | debug]
        --xml-file-prerix-digest DIGESTIBLE_PREFIX (default="")
        --xml-file-ext-digest DIGESTIBLE_EXTENSION (default=".")
        --lower-case-doc-key (lower case document key)
        --upper-case-doc-key (upper case document key)
        --fill-default-value (fill @default value in case of empty)
        --filt-in   table_name.column_name
        --filt-out  table_name.column_name:regex_pattern(|regex_pat-
tern...)
        --fill-this table_name.column_name:filling_text
        --discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
        --no-pgschema-serv (not utilize PgSchema server)
```

--pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="local-host")
--pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
--max-thrds MAX_THRDS (default is number of available proces-sors)

Appendix 8. Arguments of the **dsmerge4sphinx** main class

```
dsmerge4sphinx: Merge Sphinx data source files into one
Usage:  --xsd SCHEMA_LOCATION --dst-ds-dir DIRECTORY (default="sphin-
x_xmlpipe2") --src-ds-dir DIRECTORY (repeat until you specify all di-
rectories)
Option: --ds-name DS_NAME (default name is determined by data_source.-
conf file)
        --no-pgschema-serv (not utilize PgSchema server)
        --pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="local-
host")
        --pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
```

Appendix 9. Arguments of the **dicmerge4sphinx** main class

```
dicmerge4sphinx: Sphinx data source -> Sphinx dictionary index
Usage:  --ds-dir DIRECTORY (default="sphinx_xmlpipe2")
        --dic DIC_FILE (repeat until you specify all dictionaries)
Option: --freq FREQ_THRESHOLD (default=10)
```

Appendix 10. Arguments of the **xsd2jsonschema** main class

```
xsd2jsonschema: XML Schema -> JSON Schema conversion
Usage:  --xsd SCHEMA_LOCATION --json JSON_SCHEMA_FILE (default=stdout)
        --schema-ver JSON_SCHEMA_VER (choose from
"draft_v7" (default), "draft_v6", "draft_v4", or "latest" as
"draft_v7")
        --obj-json (use object-oriented JSON format)
        --col-json (use column-oriented JSON format, default)
        --rel-json (use relational-oriented JSON format)
Option: --no-wild-card (turn off wild card extension)
        --case-insensitive (all table and column names are lowercase)
        --field-annotation (retrieve field annotation, default)
        --no-field-annotation (do not retrieve field annotation)
        --no-cache-xsd (retrieve XML Schemata without caching)
        --json-attr-prefix ATTR_PREFIX_CODE (default="")
        --json-simple-cont-name SIMPLE_CONTENT_NAME
(default="content")
        --json-indent-offset INTEGER (default=2, min=0, max=4)
        --json-key-value-offset INTEGER (default=1, min=0, max=4)
        --json-no-linefeed (dismiss line feed code)
        --json-compact (equals to set --json-indent-offset 0 --json-
key-value-offset 0 --json-no-linefeed)
        --json-array-all (use JSON array uniformly for descendants,
effective only in column- and relational-oriented JSON format)
        --discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
```

Appendix 11. Arguments of the **xml2json** main class

```
xml2json: XML -> JSON document conversion
Usage:  --xsd SCHEMA_LOCATION --xml XML_FILE_OR_DIRECTORY --json-dir
DIRECTORY (default="json_work")
        --no-wild-card (turn off wild card extension)
        --validate (turn on XML Schema validation)
        --no-validate (turn off XML Schema validation, default)
        --well-formed (validate only whether document is well-formed)
        --xml-file-ext FILE_EXTENSION [xml (default) | gz (indicates
xml.gz suffix) | zip (indicates xml.zip suffix)]
        --schema-ver JSON_SCHEMA_VER (choose from
"draft_v7" (default), "draft_v6", "draft_v4", or "latest" as
"draft_v7")
        --obj-json (use object-oriented JSON format)
        --col-json (use column-oriented JSON format, default)
        --rel-json (use relational-oriented JSON format)
Option: --json-attr-prefix ATTR_PREFIX_CODE (default="")
        --json-simple-cont-name SIMPLE_CONTENT_NAME
(default="content")
        --json-indent-offset INTEGER (default=2, min=0, max=4)
        --json-key-value-offset INTEGER (default=1, min=0, max=4)
        --json-no-linefeed (dismiss line feed code)
        --json-compact (equals to set --json-indent-offset 0 --json-
key-value-offset 0 --json-no-linefeed)
        --json-array-all (use JSON array uniformly for descendants,
effective only in column- and relational-oriented JSON format)
        --case-insensitive (all table and column names are lowercase)
        --no-cache-xsd (retrieve XML Schemata without caching)
        --xml-file-prerix-digest DIGESTIBLE_PREFIX (default="")
        --xml-file-ext-digest DIGESTIBLE_EXTENSION (default=".")
        --lower-case-doc-key (lower case document key)
        --upper-case-doc-key (upper case document key)
        --fill-default-value (fill @default value in case of empty)
        --filt-in   table_name.column_name
        --filt-out  table_name.column_name:regex_pattern(|regex_pat-
tern...)
        --fill-this table_name.column_name:filling_text
        --discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
        --no-pgschema-serv (not utilize PgSchema server)
        --pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="local-
host")
        --pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
        --max-thrds MAX_THRDS (default is number of available proces-
sors)
```

Appendix 12. Arguments of the **xpathparser** main class

```
xpathparser: XPath 1.0 parser being aware of XML Schema
Usage:  --xsd SCHEMA_LOCATION
        --xpath-query XPATH_QUERY
        --xpath-var KEY=VALUE (repeat until you specify all variables)
        --no-rel (turn off relational model extension)
        --no-wild-card (turn off wild card extension)
        --doc-key (append document_id column in all relations, default
with relational model extension)
        --no-doc-key (remove document_id column from all relations,
effective only with relational model extension)
        --ser-key (append serial_id column in child relation of list
holder)
        --xpath-key (append xpath_id column in all relations)
Option: --case-insensitive (all table and column names are lowercase)
        --pg-public-schema (utilize "public" schema, default)
        --pg-named-schema (enable explicit named schema)
        --pg-map-big-integer (map xs:integer to BigInteger according
to the W3C rules)
        --pg-map-long-integer (map xs:integer to signed long 64 bits)
        --pg-map-integer (map xs:integer to signed int 32 bits, de-
fault)
        --no-cache-xsd (retrieve XML Schemata without caching)
        --hash-by ALGORITHM [MD2 | MD5 | SHA-1 (default) | SHA-224 |
SHA-256 | SHA-384 | SHA-512]
        --hash-size BIT_SIZE [int (32bit) | long (64bit, default) |
native (default bit of algorithm) | debug (string)]
        --ser-size BIT_SIZE [short (16bit); | int (32bit, default)]
        --doc-key-name DOC_KEY_NAME (default="document_id")
        --ser-key-name SER_KEY_NAME (default="serial_id")
        --xpath-key-name XPATH_KEY_NAME (default="xpath_id")
        --discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
        --inplace-doc-key-name INPLACE_DOCUMENT_KEY_NAME
        --doc-key-if-no-inplace (append document key if no in-place
document key, select --no-doc-key options by default)
        --no-pgschema-serv (not utilize PgSchema server)
        --pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="local-
host")
        --pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
```

Appendix 13. Arguments of the **xpath2xml** main class

```
xpath2xml: XPath 1.0 qeury evaluation to XML over PostgreSQL
Usage:  --xsd SCHEMA_LOCATION --db-name DATABASE --db-user USER --db-
pass PASSWORD (default="")
        --db-host PG_HOST_NAME (default="localhost")
        --db-port PG_PORT_NUMBER (default=5432)
        --test-ddl (perform consistency test on PostgreSQL DDL)
        --xpath-query XPATH_QUERY (repeatable)
        --xpath-var KEY=VALUE (repeat until you specify all variables)
        --out OUTPUT_FILE_OR_PATTERN (default=stdout)
        --out-dir OUTPUT_DIRECTORY
        --no-rel (turn off relational model extension)
        --no-wild-card (turn off wild card extension)
        --doc-key (append document_id column in all relations, default
with relational model extension)
        --no-doc-key (remove document_id column from all relations,
effective only with relational model extension)
        --ser-key (append serial_id column in child relation of list
holder)
        --xpath-key (append xpath_id column in all relations)
Option: --case-insensitive (all table and column names are lowercase)
        --pg-public-schema (utilize "public" schema, default)
        --pg-named-schema (enable explicit named schema)
        --pg-map-big-integer (map xs:integer to BigInteger according
to the W3C rules)
        --pg-map-long-integer (map xs:integer to signed long 64 bits)
        --pg-map-integer (map xs:integer to signed int 32 bits, de-
fault)
        --pg-map-big-decimal (map xs:decimal to BigDecimal according
to the W3C rules, default)
        --pg-map-double-decimal (map xs:decimal to double precision 64
bits)
        --pg-map-float-decimal (map xs:decimal to single precision 32
bits)
        --no-cache-xsd (retrieve XML Schemata without caching)
        --hash-by ALGORITHM [MD2 | MD5 | SHA-1 (default) | SHA-224 |
SHA-256 | SHA-384 | SHA-512]
        --hash-size BIT_SIZE [int (32bit) | long (64bit, default) |
native (default bit of algorithm) | debug (string)]
        --ser-size BIT_SIZE [short (16bit); | int (32bit, default)]
        --doc-key-name DOC_KEY_NAME (default="document_id")
        --ser-key-name SER_KEY_NAME (default="serial_id")
        --xpath-key-name XPATH_KEY_NAME (default="xpath_id")
        --discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
        --inplace-doc-key-name INPLACE_DOCUMENT_KEY_NAME
        --doc-key-if-no-inplace (append document key if no in-place
document key, select --no-doc-key options by default)
        --no-pgschema-serv (not utilize PgSchema server)
```

--pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="local-host")
--pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
--xml-no-declare (dismiss XML declaration)
--xml-no-xmlns (dismiss XML namespace declaration)
--xml-no-nil-elem (dismiss nillable element)
--xml-indent-offset INTEGER (default=2, min=0, max=4)
--xml-insert-doc-key (insert document key in result)
--xml-no-linefeed (dismiss line feed code)
--xml-compact (equals to set --xml-indent-offset 0 --xml-no-linefeed)
--verbose (verbose mode)

Appendix 14. Arguments of the **xpath2json** main class

```
xpath2json: XPath 1.0 qeury evaluation to JSON over PostgreSQL
Usage:  --xsd SCHEMA_LOCATION --db-name DATABASE --db-user USER --db-
pass PASSWORD (default="")
        --db-host PG_HOST_NAME (default="localhost")
        --db-port PG_PORT_NUMBER (default=5432)
        --test-ddl (perform consistency test on PostgreSQL DDL)
        --xpath-query XPATH_QUERY (repeatable)
        --xpath-var KEY=VALUE (repeat until you specify all variables)
        --out OUTPUT_FILE_OR_PATTERN (default=stdout)
        --out-dir OUTPUT_DIRECTORY
        --schema-ver JSON_SCHEMA_VER (choose from
"draft_v7" (default), "draft_v6", "draft_v4", or "latest" as
"draft_v7")
        --obj-json (use object-oriented JSON format)
        --col-json (use column-oriented JSON format, default)
        --no-rel (turn off relational model extension)
        --no-wild-card (turn off wild card extension)
        --doc-key (append document_id column in all relations, default
with relational model extension)
        --no-doc-key (remove document_id column from all relations,
effective only with relational model extension)
        --ser-key (append serial_id column in child relation of list
holder)
        --xpath-key (append xpath_id column in all relations)
Option: --case-insensitive (all table and column names are lowercase)
        --pg-public-schema (utilize "public" schema, default)
        --pg-named-schema (enable explicit named schema)
        --pg-map-big-integer (map xs:integer to BigInteger according
to the W3C rules)
        --pg-map-long-integer (map xs:integer to signed long 64 bits)
        --pg-map-integer (map xs:integer to signed int 32 bits, de-
fault)
        --pg-map-big-decimal (map xs:decimal to BigDecimal according
to the W3C rules, default)
        --pg-map-double-decimal (map xs:decimal to double precision 64
bits)
        --pg-map-float-decimal (map xs:decimal to single precision 32
bits)
        --no-cache-xsd (retrieve XML Schemata without caching)
        --hash-by ALGORITHM [MD2 | MD5 | SHA-1 (default) | SHA-224 |
SHA-256 | SHA-384 | SHA-512]
        --hash-size BIT_SIZE [int (32bit) | long (64bit, default) |
native (default bit of algorithm) | debug (string)]
        --ser-size BIT_SIZE [short (16bit); | int (32bit, default)]
        --doc-key-name DOC_KEY_NAME (default="document_id")
        --ser-key-name SER_KEY_NAME (default="serial_id")
        --xpath-key-name XPATH_KEY_NAME (default="xpath_id")
```

--discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
--inplace-doc-key-name INPLACE_DOCUMENT_KEY_NAME
--doc-key-if-no-inplace (append document key if no in-place document key, select --no-doc-key options by default)
--no-pgschema-serv (not utilize PgSchema server)
--pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="local-host")
--pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
--json-attr-prefix ATTR_PREFIX_CODE (default="")
--json-simple-cont-name SIMPLE_CONTENT_NAME (default="content")
--json-indent-offset INTEGER (default=2, min=0, max=4)
--json-key-value-offset INTEGER (default=1, min=0, max=4)
--json-insert-doc-key (insert document key in result)
--json-no-linefeed (dismiss line feed code)
--json-compact (equals to set --json-indent-offset 0 --json-key-value-offset 0 --json-no-linefeed)
--json-array-all (use JSON array if possible)
--verbose (verbose mode)

Appendix 15. Arguments of the **xpath2pgsql** main class

```
xpath2pgsql: Query translation from XPath 1.0 to SQL
Usage:  --xsd SCHEMA_LOCATION --db-name DATABASE --db-user USER --db-
pass PASSWORD (default="")
        --db-host PG_HOST_NAME (default="localhost")
        --db-port PG_PORT_NUMBER (default=5432)
        --test-ddl (perform consistency test on PostgreSQL DDL)
        --xpath-query XPATH_QUERY (repeatable)
        --xpath-var KEY=VALUE (repeat until you specify all variables)
        --out OUTPUT_FILE_OR_PATTERN (default=stdout)
        --out-dir OUTPUT_DIRECTORY
        --no-rel (turn off relational model extension)
        --no-wild-card (turn off wild card extension)
        --doc-key (append document_id column in all relations, default
with relational model extension)
        --no-doc-key (remove document_id column from all relations,
effective only with relational model extension)
        --ser-key (append serial_id column in child relation of list
holder)
        --xpath-key (append xpath_id column in all relations)
Option: --case-insensitive (all table and column names are lowercase)
        --pg-public-schema (utilize "public" schema, default)
        --pg-named-schema (enable explicit named schema)
        --pg-map-big-integer (map xs:integer to BigInteger according
to the W3C rules)
        --pg-map-long-integer (map xs:integer to signed long 64 bits)
        --pg-map-integer (map xs:integer to signed int 32 bits, de-
fault)
        --pg-map-big-decimal (map xs:decimal to BigDecimal according
to the W3C rules, default)
        --pg-map-double-decimal (map xs:decimal to double precision 64
bits)
        --pg-map-float-decimal (map xs:decimal to single precision 32
bits)
        --pg-tab-delimiter (use tab separated file, default)
        --pg-comma-delimiter (use comma separated file)
        --no-cache-xsd (retrieve XML Schemata without caching)
        --hash-by ALGORITHM [MD2 | MD5 | SHA-1 (default) | SHA-224 |
SHA-256 | SHA-384 | SHA-512]
        --hash-size BIT_SIZE [int (32bit) | long (64bit, default) |
native (default bit of algorithm) | debug (string)]
        --ser-size BIT_SIZE [short (16bit); | int (32bit, default)]
        --doc-key-name DOC_KEY_NAME (default="document_id")
        --ser-key-name SER_KEY_NAME (default="serial_id")
        --xpath-key-name XPATH_KEY_NAME (default="xpath_id")
        --discarded-doc-key-name DISCARDED_DOCUMENT_KEY_NAME
        --inplace-doc-key-name INPLACE_DOCUMENT_KEY_NAME
```

--doc-key-if-no-inplace (append document key if no in-place document key, select --no-doc-key options by default)
--no-pgschema-serv (not utilize PgSchema server)
--pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="local-host")
--pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
--verbose (verbose mode)

Appendix 16. Arguments of the **xmlsplitter** main class

```
xmlsplitter: Split large XML file into small ones based on XPath query
Usage:  --xsd SCHEMA_LOCATION --xml SRC_XML_FILE_OR_DIRECTORY --xml-
dir DST_DIRECTORY (default="xml_work")
        --xml-file-ext SRC_FILE_EXTENSION [xml (default) | gz (indi-
cates xml.gz suffix) | zip (indicates xml.zip suffix)]
        --xpath-doc-key XPATH_EXPR_FOR_DOC_KEY
        --no-wild-card (turn off wild card extension)
        --shard-size SHARD_SIZE (default=1)
Option: --pg-public-schema (utilize "public" schema, default)
        --pg-named-schema (enable explicit named schema)
        --no-cache-xsd (retrieve XML Schemata without caching)
        --no-pgschema-serv (not utilize PgSchema server)
        --pgschema-serv-host PG_SCHEMA_SERV_HOST_NAME (default="local-
host")
        --pgschema-serv-port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
        --verbose (verbose mode)
```

Appendix 17. Arguments of the **xmlvalidator** main class

```
xmlvalidator: Validate XML documents against XML Schema
Usage:  --xsd SCHEMA_LOCATION --xml XML_FILE_OR_DIRECTORY
        --well-formed (validate only whether document is well-formed)
        --xml-file-ext FILE_EXTENSION [xml (default) | gz (indicates
xml.gz suffix) | zip (indicates xml.zip suffix)]
Option: --sync CHECK_SUM_DIRECTORY (generate check sum files)
        --checksum-by ALGORITHM [MD2 | MD5 (default) | SHA-1 | SHA-224
| SHA-256 | SHA-384 | SHA-512]
        --max-thrds MAX_THRDS (default is number of available proces-
sors)
        --del-invalid-xml (delete invalid XML documents)
        --verbose (verbose mode)
```

Appendix 18. Arguments of the **chksumstat** main class

```
chksumstat: Report check sum directory status
Usage:  --xml XML_FILE_OR_DIRECTORY --sync-dir CHECK_SUM_DIRECTORY
        --xml-file-ext FILE_EXTENSION [xml (default) | gz (indicates
xml.gz suffix) | zip (indicates xml.zip suffix)]
Option: --checksum-by ALGORITHM [MD2 | MD5 (default) | SHA-1 | SHA-224
| SHA-256 | SHA-384 | SHA-512]
        --xml-file-prerix-digest DIGESTIBLE_PREFIX (default="")
        --xml-file-ext-digest DIGESTIBLE_EXTENSION (default=".")
        --update (update check sum files anyway)
        --max-thrds MAX_THRDS (default is number of available proces-
sors)
        --verbose (verbose mode)
```

Appendix 19. Arguments of the **pgschemaserv** main class

```
pgschemaserv: PgSchema server control
Usage:  --port PG_SCHEMA_SERV_PORT_NUMBER (default=5430)
Option: --host PG_SCHEMA_SERV_HOST_NAME (default="localhost")
        --lifetime LIFETIME_SECOND (default=1209600)
        --start (start PgSchema server, default)
        --status (report PgSchema server status)
        --stop (stop PgSchema server)
```