SchemaSpy

Graphical Database Schema Metadata Browser

- Sample Output
- FAQ
- Download
- Release Notes
- Support SchemaSpy
- John Currier

Recent Donors:
Anonymous
monocongo
chervitz

Support this project

Do you hate starting on a new project and having to try to figure out someone else's idea of a database? Or are you in QA and the developers expect you to understand all the relationships in their schema? If so then this tool's for you.

SchemaSpy is a Java-based tool (requires Java 5 or higher) that analyzes the metadata of a schema in a database and generates a visual representation of it in a browser-displayable format. It lets you click through the hierarchy of database tables via child and parent table relationships as represented by both HTML links and entity-relationship diagrams. It's also designed to help resolve the obtuse errors that a database sometimes gives related to failures due to constraints.

It is free software that is distributed under the terms of the Lesser GNU Public License 2.1. Your donations are, however, **greatly** appreciated.

If you like SchemaSpy then please vote for it on freshmeat (click rate this project), digg it and bookmark it on delicious.

SchemaSpy uses the *dot* executable from Graphviz to generate graphical representations of the table/view relationships. This was initially added for people who see things visually. Now the graphical representation of relationships is a fundamental feature of the tool. Graphvis is not required to view the output generated by SchemaSpy, but the *dot* program should be in your PATH (not CLASSPATH) when running SchemaSpy or none of the entity relationship diagrams will be generated (or use the `-gv` option).

SchemaSpy uses JDBC's database metadata extraction services to gather the majority of its information, but has to make vendor-specific SQL queries to gather some information such as the SQL associated with a view and the details of check constraints. The differences between vendors have been isolated to configuration files and are extremely limited. Almost all of the vendor-specific SQL is optional.

Browse some sample pages generated by SchemaSpy. Note that this was run against an extremely limited schema so it doesn't show the full power of the tool.

SchemaSpy is a command line tool. If you're more comfortable with the point-and-click approach then try out Joachim Uhl's SchemaSpyGUI.

SchemaSpy is now in O'Reilly's Java Power Tools book

Running SchemaSpy

You run SchemaSpy from the command line:

```
java -jar schemaSpy.jar -t dbType -db dbName [-s schema] -u user [-p password] -o outputDir
```

Commonly used parameters:

	Parameter	Description
	`-t` *`databaseType`*	Type of database (e.g. ora, db2, etc.). Use `-dbhelp` for a list of built-in types. Defaults to `ora`.
*	`-db` *`dbName`*	Name of database to connect to
*	`-u` *`user`*	Valid database user id with read access. A user id is required unless `-sso` is specified.
	`-s` *`schema`*	Database schema. This is optional if it's the same as user or isn't supported by your database. Use `-noschema` if your database thinks it supports schemas but doesn't (e.g. older versions of Informix).
	`-p` *`password`*	Password associated with that user. Defaults to no password.
*	`-o` *`outputDirectory`*	Directory to write the generated HTML/graphs to
	`-dp` *`pathToDrivers`*	Looks for drivers here before looking in driverPath in [databaseType].properties. The drivers are usually contained in .jar or .zip files and are typically provided by your database vendor.
	`-hq` `-lq`	Generate either higher or lower-quality diagrams. Various installations of Graphviz (depending on OS and/or version) will default to generating either higher or lower quality images. That is, some might not have the "lower quality" libraries and others might not have the "higher quality" libraries. Higher quality output takes longer to generate and results in significantly larger image files (which take longer to download / display), but the resultant Entity Relationship diagrams generally look better.

Parameters marked with '*' are required.

Less commonly used parameters:

`-gv` *`pathToGraphviz`*	By default SchemaSpy expects the dot executable to be in the PATH environment variable. Use this option to explicitly specify where Graphviz is installed.
`-desc "`*`Schema description`*`"`	Displays the specified textual description on summary pages. If your description includes an equals sign then escape it with a backslash. For example: `-desc "<a href\='http://schemaspy.sourceforge.net'>SchemaSpy</a>"`.
`-all`	Evaluate all schemas in a database. Generates a high-level index of the schemas evaluated and allows for traversal of cross-schema foreign key relationships. Use with `-schemaSpec "`*`schemaRegularExpression`*`"` to narrow-down the schemas to include.

`-schemas "`*`schema1,schema2`*`"`	Evaluate specified schemas. Similar to `-all`, but explicitly specifies which schema to evaluate without interrogating the database's metadata. Can be used with databases like MySQL where a database isn't composed of multiple schemas.
`-meta` *`metafile`*	*`metafile`* is either the name of an individual XML file or the directory that contains meta files. If a directory is specified then it is expected to contain files matching the pattern `[schema].meta.xml`. For databases that don't have schema substitute database for schema. See Providing Additional Metadata for details.
`-connprops` *`propsfile`* or *`key\=value;`*	Specifies additional properties to be used when connecting to the database. Either specify a .properties file (with key=value entries) or specify the entries directly, escaping the ='s with \= and separating each key\=value pair with a `;`.
`-i "`*`tableNamesRegex`*`"`	Only include matching tables/views. This is a regular expression that's used to determine which tables/views to include. For example: `-i "(.*book.*)|(library.*)"` includes only those tables/views with 'book' in their names or that start with 'library'. You might want to use `-desc` with this option to describe the subset of tables.
`-I "`*`tableNamesRegex`*`"`	Exclude matching tables/views. This regular expression excludes matching tables/views from the analysis. Can be used in conjunction with `-i`.
`-x "`*`columnNamesRegex`*`"`	Exclude matching columns from relationship analysis to simplify the generated graphs. This is a regular expression that's used to determine which columns to exclude. It must match table name, followed by a dot, followed by column name. For example: `-x "(book.isbn)|(borrower.address)"` Note that each column name regular expression must be surround by `()`'s and separated from other column names by a `|`. Excluded relationships will still show up on detail pages.
`-X "`*`columnNamesRegex`*`"`	Same as `-x` but excluded relationships will *not* show up on detail pages.
`-noviews`	Exclude all views.
`-ahic`	*A*llow *H*TML *I*n *C*omments. Any HTML embedded in comments normally gets encoded so that it's rendered as text. This option allows it to be rendered as HTML.
`-norows`	Don't query or display row counts.
`-noimplied`	Don't include implied foreign key relationships in the generated table details.
`-sso`	*S*ingle *S*ign-*O*n. Don't require a user to be specified with `-u` to simplify configuration when running in a single sign-on environment.
`-pfp`	*P*rompt *F*or *P*assword. Prompts for the password so it doesn't appear on the command line.
`-nohtml`	Only generate files needed for insertion/deletion of data (e.g. for scripts) and an XML representation of the schema.
`-loglevel`	Specifies how verbose logging of programmatic flow should be. The levels in descending order are: • `severe` (highest - least detail)

	• `warning` (default) • `info` • `config` • `fine` • `finer` • `finest` (lowest - most detail)

SchemaSpy supports many types of databases. Use `java -jar schemaSpy.jar -dbhelp` for a complete list of the built-in database types and the parameters that each one requires.
See the database types documentation if you want to add support for other types of databases or add additional functionality (e.g. to display view and check constraint SQL) to supported databases.

Type	Description
db2	IBM DB2 with 'app' Driver
db2net	IBM DB2 with 'net' Driver
udbt4	DB2 UDB Type 4 Driver
db2zos	DB2 for z/OS
derby	Derby (JavaDB) Embedded Server
derbynet	Derby (JavaDB) Network Server
firebird	Firebird
hsqldb	HSQLDB Server
informix	Informix
maxdb	MaxDB
mssql	Microsoft SQL Server
mssql05	Microsoft SQL Server 2005
mssql-jtds	Microsoft SQL Server with jTDS Driver
mssql05-jtds	Microsoft SQL Server 2005 with jTDS Driver
mysql	MySQL
ora	Oracle with OCI8 Driver
orathin	Oracle with Thin Driver
pgsql	PostgreSQL
sqlite	SQLite
sybase	Sybase Server with JDBC3 Driver
sybase2	Sybase Server with JDBC2 Driver
teradata	Teradata (requires -connprops)

A MySQL example:

```
 java -jar schemaSpy.jar -t mysql -o library -host localhost -db library -u user -p password
```

will create a series of files in the `library` directory that give the details of the schema in the database `library`. This is what I used to generate the sample output.

An MS SQL Server example:

```
 java -jar schemaSpy.jar -t mssql -db library -host localhost -port 1433 -u user -p password -o
library
```

does the same thing as the MySQL example, but specifies an `mssql` database type with MS SQL Server-specific database connection parameters.

Providing Additional Metadata

Metafiles are XML-based files that provide additional metadata about the schema being evaluated. See the `-meta` parameter. Here are some of the things that you can define in this XML:

- Schema comments
- Table comments
- Primary keys
- Foreign keys
- Cross-schema foreign keys
- Disabled implied relationships
- Disabled diagram assocations
- etc.

The XML schema that defines the structure of these files is available here. There are also some sample XML files (a work in progress) that were used to generate these pages. Note that this group of MySQL databases had almost no foreign key relationships defined.

Some information about the developer, John Currier, is available here.
Feedback on problems and/or enhancements is appreciated.