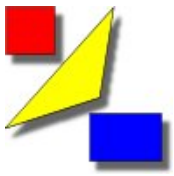
# *Schema Version Control for Oracle (SVCO)*

*Version: 1.2.2*

*End-User Documentation*



SUMsoft Solutions

# Introduction

## What Is Version control or Revision control?

Revision control (also known as version control (system) (VCS), source control or (source) code management (SCM)) is the management of multiple revisions of the same unit of information. Version control is the process of managing and tracking changes, and is essential regardless of the size of your organization or the type of software you develop.
Generally speaking, version control tools provide these features:

- Central repository
  Source control tools provide a central repository to host all files, including source code, executables, documentations, and other files such as images. Project team members can access the central location from their own workstation. This central repository serves as the "official" copy of the project files.

- Change tracking
  Version control tools keep track of the changes made in a file when it's versioned. This allows developers to reconstruct earlier versions of the file, which is useful for recovering earlier work or to get to an earlier release.

- File difference comparison
  Version control tools allow users to compare different versions of the file in its repository, or to compare the file in repository against the (local) file that the user is currently working on.

- History
  Developers can examine the revision history for each file, including the comments made by the developer when revision was created.

- Label files
  Version control tools allow developer to mark a set of files with a label. This is used to identify all files in a release. Users can retrieve the whole set by the label name.

## Using SVCO

The great advantage of SVCO is that it's fully integrated solution. The repository and all SVCO functionality persist inside Oracle. You don't need any extra version control system and you should not give up to your favourite Oracle development tools. On the other side you could easily integrate SVCO in your own development process.
After installation you have a new user SVCO and two roles to regulate SVCO functionality and repository access. *SVCO user* is the central repository of our system. This user or schema will store all information about schema objects and their differences. Normally you will never work as *SVCO user* because all SVCO functionality is granted to both roles.

### SVCO roles concept

There are 2 roles available to regulate SVCO functionality and repository access. Initially both roles are granted to SYS and SVCO users only. The role SVCO_REP_OPERATOR gives you possibility to execute any procedures from REPOSITORY package. All users which get SVCO_REP_OPERATOR granted also have the read only access to the repository options and repository views. Such users are able to generate revisions only for they own schemas. The second role SVCO_REP_ADMIN includes all privileges of SVCO_REP_OPERATOR role. With SVCO_REP_ADMIN role you could also execute any procedures from REPOSITORY_ADMIN package. As well you have full access to the repository options. But the main difference is that with SVCO_REP_ADMIN role you could create revisions for any schemas.

# SVCO functionality

## Creating new schema revision

Each object in the repository is held as a set of snapshots, called *object versions*, of its state at certain points in time. The object versions are linked together by *associations* that record how each object version evolved from its predecessors. This web of associations, represented by the version tree, is called the *version history* of the object. Creating new schema revision is a process of making a snapshot of all schema objects. Every revision can have:

| | | |
|---|---|---|
| Revision Identifier | View SVCO.REPOSITORY_LOG.REVISION | *System given positive NUMBER. Start with 1* |
| Schema Name | View SVCO.REPOSITORY_LOG.SCHEMANAME | *Username which revision belongs to.* |
| Version | View SVCO.REPOSITORY_LOG.VERSION | *Your own identifier for revision. Optional.* |
| Comments | View SVCO.REPOSITORY_LOG.COMMENTS | *Your own comments for revision. Optional.* |
| Created date & time | View SVCO.REPOSITORY_LOG.CREATED | *System given DATE when the revision was created.* |

To create a new revision just call procedure *CreateRevision* from REPOSITORY package.

### Generating DDL script from revision

After revision is created you could generate DDL script from complete revision or only from certain objects. Procedure *CreateDDLFromRevision* from REPOSITORY_ADMIN package performs this operation.

### Revisions difference comparison

Call procedure *GetRevisionsDifference* from REPOSITORY package to obtain the list of objects changed in both revisions regards to the same schema.

### Delete the last revision

Only the last (most actual) revision could be deleted. To delete the last revision you should call procedure *DeleteLastRevision* from REPOSITORY_ADMIN package.

## SVCO repository options

Certain behaviour of SVCO could be managed with repository options. All such options are located in the table REPOSITORY_OPTIONS.

| *OPTION NAME* | *DEFAULT VALUE* | *COMMENTS* |
|---|---|---|
| COMPRESS | TRUE | If TRUE, compress all objects data in the repository to save disk space. |
| COMPRESS_QUALITY | 6 | Speed versus efficiency of resulting compressed output. Valid values are the range 1..9, with a default value of 6. 1=fastest compression, 9=slowest compression and best compressed size. |
| ENCRYPT | FALSE | If TRUE, encrypt all objects data in the repository. |

| | | |
|---|---|---|
| IGNORE_OBJECTS | | Ignore objects from revision. Format is SCHEMA_NAME.OBJECT_NAME Wildcards in object name are allowed. More than one option is possible. Default value is always ignored. Use # to escape wildcards e.g. TEST.T#_% |
| INDEX_SEGMENT_ATTRIBUTES | TRUE | If TRUE, emit segment attributes (physical attributes, storage attributes, tablespace, logging. |
| INDEX_STORAGE | TRUE | If TRUE, emit storage clause. (Ignored if INDEX_SEGMENT_ATTRIBUTES is FALSE.) |
| INDEX_TABLESPACE | TRUE | If TRUE, emit tablespace. (Ignored if INDEX_SEGMENT_ATTRIBUTES is FALSE.) |
| SEQ_IGNORE_NEXTVAL | TRUE | if TRUE, ignore last sequence value (NEXTVAL) |
| SYSTEM_GENERATED | FALSE | If TRUE, select indexes or triggers even if they are system-generated. If FALSE, omit system-generated indexes or triggers. Defaults to FALSE. |
| TABLE_CONSTRAINTS | TRUE | If TRUE, emit all non-referential table constraints. |
| TABLE_OID | FALSE | If TRUE, emit the OID clause for object tables. |
| TABLE_REF_CONSTRAINTS | TRUE | If TRUE, emit all referential |

| | | |
|---|---|---|
| | | constraints (foreign keys) |
| TABLE_SEGMENT_ATTRIBUTES | TRUE | If TRUE, emit segment attributes (physical attributes, storage attributes, tablespace, logging. |
| TABLE_STORAGE | TRUE | If TRUE, emit storage clause. (Ignored if TABLE_SEGMENT_ATTRIBUTES is FALSE.) |
| TABLE_TABLESPACE | TRUE | If TRUE, emit tablespace. (Ignored if TABLE_SEGMENT_ATTRIBUTES is FALSE.) |
| TYPE_OID | FALSE | If TRUE, emit the OID clause. |
| VIEW_FORCE | TRUE | If TRUE, use the FORCE keyword in the CREATE VIEW statement. |
| INCLUDE_OBJECTS | | Include objects to revision. Format is SCHEMA_NAME.OBJECT_NAME Wildcards in object name are allowed. More than one option is possible. Default value is always ignored. Use # to escape wildcards e.g. TEST.T#_% |

## SVCO repository views

View REPOSITORY_LOG shows all schema revisions located in SVCO repository.

| *VIEW COLUMN* | *COMMENTS* |
|---|---|
| SCHEMANAME | Schema (user owner) name |
| REVISION | Revision Number |
| VERSION | External/Custom Version |

| | |
|---|---|
| CREATED | When revision was created |
| COMMENTS | User comments to revision |

View REPOSITORY_OBJECTS_LOG consists of all versioned objects.

| *VIEW COLUMN* | *COMMENTS* |
|---|---|
| SCHEMANAME | Schema (user owner) name |
| REVISION | Revision Number |
| OBJECTTYPE | See "Supported object types" |
| OBJECTNAME | The name of versioned object |
| OBJECTSTATUS | 'New' – new added object<br>'Changed' – the object changed in this revision<br>'Deleted' – the object was deleted in this revision<br>'Referenced' – the object was not modified in this revision, see next columns |
| REFERENCESCHEMANAME | For objects in status 'Referenced' – used to identify the appropriate revision number |
| REFERENCEREVISION | For objects in status 'Referenced' – last operation with the object was in that revision |

The combination of columns SCHEMANAME and REVISION is unique and exactly identifies the schema revision. To get the list of all objects inside schema revision you could join both views by SCHEMANAME and REVISION columns.

## Supported object types

Current version of SVCO supports the following Oracle schema object types:

- LIBRARY
- TYPE (spec and body)
- DB_LINK
- CLUSTER
- TABLE
- TRIGGER
- INDEX
- INDEXTYPE
- FUNCTION
- PROCEDURE
- VIEW
- MATERIALIZED VIEW
- MATERIALIZED VIEW LOG
- SEQUENCE
- SYNONYM
- PACKAGE (spec and body)
- COMMENT
- OBJECT_GRANT

Other object types are planed for the next SVCO release.

*We would appreciate any comments and suggestions concerning the product.*