|  |  |
| --- | --- |
| Subject: | Manual SQLXport tool |
| Date: | June 18, 2023 |
| Version: | 1.0.0.0 |

**About**

This application is much like standard ‘export’ and ‘import’ tools. E.g. like Oracle’s “exp” and “imp” programs, but the difference is that this application need not be run on the database server machine itself, but can be used through an ODBC driver.

**But BEWARE!:**

This program is \*\*\*NOT\*\*\* a complete import or export program as supported by your RDBMS brand.

It only supports the exporting and importing of a single schema, and then only the parts I was using in day-to-day programs

**DISCLAIMER**

This program has only be tested with the Oracle and Microsoft SQL-Server databases. For other RDBMS vendor databases, you are ‘on-your-own’ !!

**What IS processed**

in both export and import mode, the following objects and properties are processed:

- Tables and column definitions

- Data rows in tables

- Indices of tables (column based and function based indices)

- Primary key constraints

- Foreign key constraints

- Check constraints on tables (not null, check etc)

- Unique constraints on tables (through unique indices)

- Views

- Sequences needed for primary keys

- Stored procedures

- Stored functions

- Triggers on tables

- Private synonyms for tables/views/procedures/functions/sequences

- Privileges on objects (tables, views, procedures etc)

**What is NOT being processed**

and where you need the ‘real’ export/import tools

- Database links

- Tablespace information

- Dimensions

- Directory / Disk-groups

- Flashback archives

- Editions

- Java definitions

- Jobs for the queues

- Materialized views

- Objects in the recycle bin

- Partities en clusters voor tabellen of indices

**The name**

It is possible to rename the executable on the file system to “export.exe” or “import.exe”. This will lead the program to believe that the default direction option is either ‘export’ or ‘import’.

**The command structure**

on the command line is as follows:

SQLXport /DIR:{export | import} options

The option “/DIR:” is always needed, unless you have renamed the executable as described above.

This table gives all the other possible options:

|  |  |
| --- | --- |
| Option | Explanation |
| /DBASE:<database> | Name of the database. In fact the ODBC datasource connection name. So not the physical name of the database. |
| /USER:<user> | Name of the user that will log in to the database and perform the export or import. |
| /PASSWD:<password> | Very secret!! |
| /SCHEMA:<schema> | Name of the schema that will be exported or imported. To be able to import a schema, it must already exist prior to starting this program. |
| /FILE:<bestand> | Filename that will be used to export to / import from. If not explicitly stated the default name “ExportDump.dmp” will be used. |
| /OBJECT:<naam> | Optional name of the object to export to a dump file, without touching any other object.  You can also use all the characters from the “LIKE” condition, to form patterns of objects. |
| /FILTER:<filter> | Extra filter (WHERE statement) to select rows from a table. Does only work in the ‘export’ mode. The keyword ‘where’ itself should NOT be included. |
| /PARAMS:<filename> | Filename with all parameters from this table. |
| /WAIT:<seconds> | Number of seconds to wait while blocking tables with the /CONSISTENT option. |
| /COMMIT:<n> | Does a ‘COMMIT’ after importing <n> records. Default is 1000. |
| /CONSISTENT | All tables will be blocked in share mode before the export starts. Datarows are therefore guaranteed to be consistent over foreign key constraints. |
| /DROP | Drop all objects from the schema to be imported. The schema itself is not removed. This allows for a ‘clean’ import. |
| /LIST | Make a list of all SQL statements from the dump file. Can be used without a database connection to see what’s in the database dump file. |
| /DEBUG | Shows the data rows for the table in the logfile. Mind you: the logfile can become very BIG! |
| /STATISTICS | Re-calculate database statistics after import of all the tables. |
| /RECOMPILE | Does a recompile of all invalid objects in the schema after the import. |
| /STRIPDIAC | Remove diacritics from ISO8859 characters (only at import) |
| /NOGRANTS | Do NOT export privileges on the objects |
| /NOROWS | Do NOT export the data rows of the tables |
| /NOCONSTRAINTS | Do NOT export constraints like primary/foreign keys and unique keys |
| /NOSOURCE | Do NOT export stored procedures and functions |
| /CREATESQL | Only create an export with the structure (no data), so we can generate an empty new database at import. |
| /NOHANG | Do not wait for the regular ending of the program (in case of a Virus Scanner) but force the ending of the program. |
| /HELP of /? | Help page with all the information in this table |

**Preparation**

In order to get the correct working of the program, it is absolutely necessary that the user that will log in to perform the export or import has the correct rights to access all objects that you want to export.

This seems like a rather ‘lame’ requirement, but it is often forgotten.

Most database systems have special super user accounts (admin, sys, system, etc) that have these rights. And otherwise there will exist a special type of role (e.g.. FULL EXPORT RIGHTS) that was designed to do precisely this!

**The logfile**

Every export or import will not only show the proceedings on the terminal via standard-output, but also writes the results to a logfile, in case we cannot keep up with the terminal screen.

This logfile is written under the same name as the dump file, but the extension has been replaced by the “\_export.txt” or “\_import.txt” string, dependent on the action we take.

**Example:**

Here is a small example:

C:\tmp>sqlxport /dir:export /user:inform /passwd:secret /schema:inform /dbase:firstdb /file:firstdb\_inform.dmp

This results in an export file with the complete “inform” schema in it.

Here are some reverse actions:

C:\tmp>sqlxport /dir:import /file:firstdb\_inform.dmp /list

C:\tmp>sqlxport /dir:import /user:inform /passwd:secret /schema:inform /dbase:seconddb /drop /file:firstdb\_inform.dmp

The first command wilt render a list in “firstdb\_inform\_import.txt”, while the second command will take care of the ‘real’ physical import of the file in the database ‘seconddb’, erasing (dropping) the original schema.