<https://cat.totvs.com.br/eConsultor/Pesquisa/kbase_Det.aspx?kbid=41072>

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Kbase 41072: Procedimentos de dump/load dos bancos emsdev, finance e payroll (SQL-92) | | | Autor | Gionei Mistura - CAT | | Acesso | Público | | Publicação | 28/08/2013 | |

|  |
| --- |
| Procedimentos de dump/load dos bancos emsdev, finance e payroll (SQL-92):  1 - Realizar o dump das definições através do SQLSCHEMA:  sqlschema -o emsdev.dfsql -t %.% -u pub -a pub progress:T:corp-gioneimi:15000:emsdev  2 - Efetuar o dump dos dados através do utilitário SQLDUMP:  sqldump -t %.% -u pub -a pub progress:T:corp-gioneimi:15000:emsdev  3 - Criar o banco destino (neste caso, o emsdev);  4 - Criar os usuários "sysprogress" e "pub" em cada um dos bancos. Em seguida, conectar com o usuário  "sysprogress" através do sqlexp e dar permissão de "resource" para o usuário "pub":  grant resource to pub; commit;  5 - Efetuar o load das definições:  sqlexp -user pub -password pub -db emsdev -H corp-gioneimi -S 15000 -infile emsdev.dfsql  6 - Efetuar o load dos dados. Neste ponto, as tabelas devem ser lidas uma a uma. Para facilitar, no Windows pode-se  utilizar o comando "for":  for %i in (\*.dsql) do sqlload -t %~ni -u pub -a pub progress:T:localhost:15000:emsdev  7 - Exportar as sequencias via Data Admistration do banco origem e importá-las no destino. |

DUMP

<https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/dmadm%2Fsqldump-utility.html%23>

**SQLDUMP utility**

A command-line utility that dumps application data from SQL tables into one or more files.

**Syntax**

|  |
| --- |
| sqldump -u *user\_name***[** -a *password***][** -C *code\_page\_name***]**         -t **[***owner\_name*.**]***table\_name1* **[[**,*owner\_name*.**]***table\_name2*,**...]**         **[** -n *tenant\_name1,***...]**         *db\_name* |

**Parameters**

-u*user\_name*

Specifies the user id SQLDUMP used to connect to the database. If you omit the user\_name and password parameter values, SQLDUMP prompts you for the values. If you omit user\_name and supply a password, SQLDUMP uses the value defined in the USER environment variable as the user\_name value.

-a *password*

Specifies the password used by the database for authentication.

-C *code\_page\_name*

A case-insensitive character string that specifies the name of the dump file's code page. If the -C parameter specifies a code page name that is not valid, the utility reports a run-time error. If the -C parameter does not appear at all, the code page name defaults to the client's internal code page:

*If set, the value of the client's SQL\_CLIENT\_CHARSET environment variable

*If not set, the name of the code page of the client's locale

For example, you might use the -C parameter to have a Windows client using the MS1250 code page produce a dump file using the ISO8859-2 code page (to read later on a UNIX machine, perhaps). Although you can accomplish this by setting the client's SQL\_CLIENT\_CHARSET environment variable, using the -C parameter might be easier.

-t *owner*\_*name.table*\_*name*

Specifies a list of one or more tables to dump to a file. This parameter is required. Pattern matching is supported in both*owner\_name*and *table\_name*, using a percent sign (%) for one or more characters and an underscore (\_) for a single character. The pattern matching follows the standard defined by the LIKE predicate in SQL.

You can dump a single table, a set of tables, or all tables. If you omit the optional *owner\_name* qualifier, SQLDUMP uses the name specified by the -u parameter.

-n *tenant\_name*

Specifies a list of one or more tenants to dump to a file. This parameter is optional.

*db*\_*name*

Specifies the database where you are dumping tables. You can dump tables from one database each time you invoke SQLDUMP. There is no option flag preceding the *db\_name*. This parameter is required and must be the last parameter specified. The database name is specified in the following way: progress:T:*localhost*:*demosv*:*jo*.

SQLDUMP dumps application data from SQL tables into one or more files. You can load the data from the files into another database with the SQLLOAD utility. The SQLDUMP utility does not dump data from ABL tables.

The SQLDUMP utility writes user data in row order into ASCII records with variable-length format. The column order in the files is identical to the column order in the tables. The utility writes both format and content header records to the dump file. You can dump multiple tables in a single execution by specifying multiple table names, separated by commas. Make sure there are no spaces before or after commas in the table list.

Data for one table always goes to a single dump file. Each dump file corresponds to one database table. For example, if you specify 200 tables in the SQLDUMP command, you will create 200 dump files. The SQLDUMP utility assigns the filenames that correspond to the *owner\_name* and *table\_name* in the database, with the file extension .dsql. If a dump file for a specified table already exists, it will be overwritten and replaced. Dump files are created in the current working directory.

The format of the records in a dump file is similar to the ABL .d file format:

*Converts all values to character representation

*Delimits CHARACTER values with double quotes

*Can contain any embedded characters except for NULL values, allowing commas, newlines, and other control characters

*Uses two sets of double quotes to escape embedded double quotes

*Delimits NUMERIC and other non-character data types using a space

*Processes TIMESTAMP data as if it were CHARACTER data

*Has a size limit of 2K for a single column value

*Has a maximum record length of 32K for dump file records

Any error is a fatal error, and SQLDUMP halts the dumping process so that data integrity will not be compromised. SQLDUMP reports errors to standard output.

After successful processing, SQLDUMP writes a summary report to standard output. For each table SQLDUMP processes, the report shows:

*Table name

*Dump filename

*Number of records dumped

*Number of bytes dumped

*Number of seconds required for processing

**Example**

This example directs the SQLDUMP utility to write the data from two tables to two dump files. The *user\_name* and *password*for connecting to the database are tucker and sulky. The tucker account must have the authority to access the customers and products tables in database salesdb with*owner\_name*martin, as shown:

|  |
| --- |
| sqldump -u tucker -a sulky -t martin.customers,martin.products progress:T:thunder:4077:salesdb |

This example directs the SQLDUMP utility to write the data from all tables in the salesdb database that begin with any of these strings: cust, invent, and sales, and having any owner name that the user tucker has authority to access. The *user\_name* and *password*for connecting to the database are tucker and sulky, as shown:

|  |
| --- |
| sqldump -u tucker -a sulky -t%.cust%,%.invent%,%.sales% progress:T:thunder:4077:salesdb |

This example directs the SQLDUMP utility to write the data from all tables for all owner names in the salesdb database:

|  |
| --- |
| sqldump -u tucker -a sulky -t %.% progress:T:thunder:4077:salesdb |

This example directs the SQLDUMP utility to dump the data from the tenants ten1 and ten2 to two SQL dump files respectively. The *user\_name* and *password* to connect to the database are supertenUser@superdom and superten. The supertenUser account in the superdom domain must have the authority to access the ten1 and ten2 tenant tables in mtdb database.

To separate the tenant specific data, SQLDUMPutility creates separate directories for each tenant. The ten1/<OWNER>.MTTAB1.DSQL and ten2/<OWNER>.MTTAB1.DSQL are the two directories that are created to dump tenant data:

|  |
| --- |
| sqldump -u supertenUser@superdom -a superten -t mttab1 -n ten1, ten2 progress:T:localhost:9999:mtdb |

If regTenantUser is mapped to a regular tenant, then this example directs the SQLDUMP utility to dump the data for the regTenantUser tenant's partition:

|  |
| --- |
| sqldump -u regTenantUser@OpenEdgeA -a regTenant -t mttab1 progress:T:localhost:9999:mtdb |

If dbaUser is mapped to a DBA, then this example directs the SQLDUMP utility to dump the tenant-specific data for all the tenants in their respective directory:

|  |
| --- |
| sqldump -u dbaUser -a dba -t mttab1 progress:T:localhost:9999:mtdb |

If superten is mapped to a super-tenant, then this example directs the SQLDUMP utility to dump all the tenants which start with the word ‘ten' for the table mttab:

|  |
| --- |
| sqldump -u superTenUser@superdom -a superten -t mttab1 -n ten% progress:T:localhost:9999:mtdb |

**Notes**

*The *db\_name* must be the last parameter given.

*Before you can run SQLDUMP against a database server, the server must be configured to accept SQL connections and must be running.

*Each dump file records character set information in the identifier section of each file. For example:

|  |
| --- |
| A^B^CProgress     sqlschema       v1.0       Quote fmt A^B^CTimestamp    1999-10-19   19:06:49:0000 A^B^CDatabase     dumpdb.db A^B^CProgress Character Set: iso8859-1 A^B^CJava Charcter Set: Unicode UTF-8 A^B^CDate Format: MM/DD/YYYY |

The character set recorded in the dump file is the client character set. The default character set for all non-JDBC clients is taken from the local operating system through the operating system apis. JDBC clients use the Unicode UTF-8 character set.

To use a character set different than that used by the operating system, set the SQL\_CLIENT\_CHARSET environment variable to the name of the preferred character set. You can define any OpenEdge supported character set name. The name is not case sensitive.

*SQLDUMP does not support the following characters in schema names:

*Double quote (")

*Forward slash (/)

*Backslash (\)

*SQLDUMP supports schema names that contain special characters such as, a blank space, a hyphen (-), or pound sign (#). These names must be used as delimited identifiers. Therefore, when specifying names with special characters on a UNIX command line, follow these rules:

*Use double quotes to delimit identifiers.

*So that the command line does not strip the quotes, use a backslash (\) to escape the double quotes used for delimited identifiers.

*Use double quotes to enclose any names with embedded spaces, commas, or characters special to a command shell (such as the Bourne shell). This use of quotes is in addition to quoting delimited identifiers.

For example, to dump the table Yearly Profits, use the following UNIX command-line:

|  |
| --- |
| sqldump -t "\"Yearly Profits\"" -u **xxx** -a *yyydb\_name* |

*In Windows, the command interpreter rules for the use of double quotation marks varies from UNIX.

*By default, SQLDUMP displays promsgs messages using the code page corresponding to *code\_page\_name*. That is, if you are dumping a Russian database, and *code\_page\_name* specifies the name of a Russian code page, the client displays promsgs messages using the Russian code-page, (unless you specify a different code page by setting the client's SQL\_CLIENT\_CHARSET\_PROMSGS environment variable).

LOAD

<https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/dmadm%2Fsqlload-utility.html%23>

# SQLLOAD utility

A command-line utility that loads user data from a formatted file into an SQL database.

## Syntax

|  |
| --- |
| sqlload -u *user\_name***[** -a *password***][** -C **code\_page\_name]**       -t **[***owner\_name*.**]***table\_name1***[[**,*owner\_name*.**]***table\_name2*, ...**]**        **[** -l *log\_file\_name***][** -b *badfile\_name***][** -e *max\_errors***]**        **[** -F comma **|** quote **][** -c *commit\_frequency***]**        **[** -n *tenant\_name1,***...]**        *db\_name* |

## Parameters

u *user\_name*

Specifies the user SQLLOAD uses to connect to the database. If you omit the *user\_name* and *password*, SQLLOAD prompts you for these parameter values. If you omit the *user\_name* and supply a password, SQLLOAD uses the value defined in the USER environment variable.

-a *password*

Specifies the *password* used by the database for authentication.

-C *code\_page\_name*

A case-insensitive character string that specifies the name of the dump file's code page. If the -C parameter specifies a code page name that is not valid, a run-time error is reported. If the -C parameter does not appear at all, the code page name defaults to the client's internal code page:

*If set, the value of the client's SQL\_CLIENT\_CHARSET environment variable

*If not set, the name of the code page of the client's locale

For example, you might use the -C parameter to load a dump file whose code page is ISO8859-2, using a Windows client whose code page is MS1250. Although you can accomplish this by setting the client's SQL\_CLIENT\_CHARSET environment variable, using the -C parameter might be easier.

-t *owner*\_*name.table*\_*name*

Specifies a list of one or more tables to load into a database. This parameter is required. Pattern matching is supported, using a percent sign (%) for multiple characters and an underscore (\_) for a single character. The pattern matching follows the standard for the LIKE predicate in SQL. You can load a single table, a set of tables, or all tables. If you omit the optional *owner\_name* table qualifier, SQLLOAD uses the name specified by the -u parameter. The files from which SQLLOAD loads data are not specified in the SQLLOAD syntax. The utility requires that the filename follow the naming convention *owner\_name*.*table\_name*.dsql.

-l *log\_file\_name*

Specifies the file to which SQLLOAD writes errors and statistics. The default is standard output.

-b *badfile\_name*

Specifies the file where SQLLOAD writes rows that were not loaded.

-e *max\_errors*

Specifies the maximum number of errors that SQLLOAD allows before term processing. The default is 50.

-F comma **|** quote

Directs SQLLOAD to load data in comma-delimited format or quote-delimited format. The default is quote.

-c *commit\_frequency*

Specifies the number of records written to the database before a commit is performed. Committed records cannot be rolled back if an error occurs during the load.

-n *tenant\_name*

Specifies a list of one or more tenants to dump to a file. This parameter is optional.

*db*\_*name*

Identifies the database where you are loading tables. You can load tables into a single database each time you invoke SQLLOAD. There is no option flag preceding the *db\_name*. This parameter is required, and must be the last parameter specified. The database name is specified in the following way: progress:T:*localhost*:*demosv*:*jo*.

SQLLOAD loads user data from a formatted file into an SQL database. Typically, the source file for the load is created by executing the SQLDUMP utility. The SQLLOAD utility can process a source file created by another application or utility, if the format of the file conforms to SQLLOAD requirements. The file extension made available to SQLLOAD for processing must be .dsql. See the entry on SQLDUMP for a description of the required file format.

The SQLLOAD utility reads application data from variable-length text-formatted files and writes the data into the specified database. The column order is identical to the table column order. SQLLOAD reads format and content header records from the dump file. You can load multiple tables in a single execution by specifying multiple table names, separated by commas. Data for one table is from a single dump file. Every source file corresponds to one database table. For example, if you specify 200 tables in the SQLLOAD command, you will load 200 database tables.

The format of the records in the input files is similar to the ABL .d file dump format. See [**SQLDUMP utility**](https://documentation.progress.com/output/ua/OpenEdge_latest/dmadm/sqldump-utility.html#wwID0EXSG2) for a description of the record format. The maximum record length SQLLOAD can process is 32K.

Each database record read is share-locked for consistency. You must ensure that the SQL Server has a lock table large enough to contain one lock for every record in the table. The default lock table size is 10,000 locks.

SQLLOAD writes any errors to standard output and halts the loading process for any error so that data integrity is not compromised.

## Example

This example directs the SQLLOAD utility to load the data from two dump files into the salesdb database. The input files to SQLLOAD must be tucker.customers.dsql and tucker.products.dsql, as shown:

|  |
| --- |
| sqlload -u tucker -a sulky -t tucker.customers,tucker.products progress:T:thunder:4077:salesdb |

This example directs SQLLOAD to load the data from all appropriately named dump files into the specified tables in the salesdb database:

|  |
| --- |
| sqlload -u tucker -a sulky -t %.cust%,%.invent%,%.sales% progress:T:thunder:4077:salesdb |

This example illustrates the creation of tenant directories when the tenants are dumped to a formatted file using SQLLOADutility:

|  |
| --- |
| sqlload -u supertenUser@superdom -a superten -t mttab1 -n ten1, ten2  progress:T:localhost:9999:mtdb |

If the regTenant is mapped to a regular tenant, then this example directs the SQLLOAD utility to load the data of the partition of the tenant to mttab1 table:

|  |
| --- |
| sqlload -u regTenantUser@OpenEdgeA -a regTenant -t mttab1  progress:T:localhost:9999:mtdb |

If the dbaUser is a DBA, then this example directs the SQLLOAD utility to load the tenant-specific data for all the tenants in their respective directories:

|  |
| --- |
| sqlload -u tucker -a sulky -t %.cust%,%.invent%,%.sales% progress:T:thunder:4077:salesdb |

If the superTenUser is a super-tenant, then this example directs the SQLLOAD utility to load all the data associated with the tenants which start with the word ‘ten' of the table mttab1:

|  |
| --- |
| sqlload -u superTenUser@superdom -a superten -t mttab1 -n ten%  progress:T:localhost:9999:mtdb |

If the dbaUser is a DBA, then this example directs the SQLLOAD utility to load all the data for tenants for the table mttab1:

|  |
| --- |
| sqlload -u superTenUser@superdom -a superten -t mttab1 -n ten%  progress:T:localhost:9999:mtdb |

## Notes

*The db\_name must be the last parameter given.

*Before you can run SQLLOAD against a database server, the server must be configured to accept SQL connections and must be running.

*The character set used by SQLLOAD must match the character set information recorded in each dump file. If the character sets do not match, the load is rejected. You can use the SQL\_CLIENT\_CHARSET environment variable to specify a character set.

Each dump file you create with SQLDUMP contains character set information about that file. The character set recorded in the dump file is the client character set. The default character set for all non-JDBC clients is taken from the local operating system through the operating system APIs. JDBC clients use the Unicode UTF-8 character set.

To use a character set different than that used by the operating system, set the SQL\_CLIENT\_CHARSET environment variable to the name of the preferred character set. You can define any OpenEdge-supported character set name. The name is not case sensitive.

*At run time, SQLLOAD reports an error if it detects a mismatch between the code page of the dump file being loaded and the code page of the client running SQLLOAD.

*By default, SQLLOAD displays promsgs messages using the code page corresponding to code\_page\_name. That is, if you are restoring a Russian database and *code\_page\_name* specifies the name of a Russian code page, the client displays promsgs messages using the Russian code-page (unless you specify a different code page by setting the client's SQL\_CLIENT\_CHARSET\_PROMSGS environment variable).

*SQLLOAD does not support the following characters in schema names:

*Double quote (")

*Forward slash (/)

*Backslash (\)

*SQLLOAD supports schema names that contain special characters, such as a blank space, a hyphen (-), or pound sign (#). These names must be used as delimited identifiers. Therefore, when specifying names with special characters on a UNIX command line, follow these rules:

*Use double quotes to delimit identifiers.

*Use a backslash (\) to escape the double quotes used for delimited identifiers.

*Use double quotes to enclose any names with embedded spaces, commas, or characters special to a command shell (such as the Bourne shell). This use of quotes is in addition to quoting delimited identifiers.

For example, to load the table Yearly Profits, use the following UNIX command-line:

|  |
| --- |
| sqlload -u *xxx* -a *yyy* -t "\"Yearly Profits\"" *db\_name* |

*In Windows, the command interpreter rules for the use of double quotation marks varies from UNIX.