SQL Data Synchronization Facility

Software
Product
Research

SQL/Data Synchronization Facility for VSE
Version 1.1

Table of Contents

1 Functional Description

The **SQL Data Synchronization Facility** ("**SQL/DSF"**) assists DB2/VSE database administration personnel in managing **distributed** database environments. In such environments, the same data is often present at different locations in different databases. Infocenters for example, usually do not directly access the operational database tables but a copy of them. Moreover, the individual "nodes" of the distributed database may run different platforms of IBM's DB/2 RDBMS and interconnect by means of IBM's **DRDA** protocol.

In distributed database environments, it must be ensured that the distributed copies of the data remain in a consistent state. The facilities offered by SQL/DSF help database administrators to achieve this goal.

SQL/DSF offers the following functions:

Table Synchronization
Applies modifications on tables in a DB2/VSE database to tables in any other database of the DB2 family.

Table Transfer
Copies entire tables from a database of the DB2 family, to tables in another DB2 database.

Table Compare
Compares tables in a database of the DB2 family, with tables in another DB2 database.

Conditional Table Transfer
Compares tables in a database of the DB2 family, with tables in another DB2 database. If the compare fails, the function initiates a table transfer from the source to the target database.

These functions are invoked:

- Using VSE JCL.
- By inserting the function request in the **SQLDSFS RULES** file of the SQL/DSF Scheduler.[1]

[1]The Scheduler is a VSE partition that runs the **SQLDSFS** program, which is part of the program product.

1.1 Table Synchronization

Table Synchronization applies the data modifications performed on a table in a DB2/VSE database to the same table in another database. Synchronization is a cost-effective alternative for a complete table copy, since synchronization will apply only the INSERT, DELETE and UPDATE commands executed during a DB2/VSE session on the source table.

The synchronization function runs in the VSE environment. The source table **must** be a table in a DB2/VSE database. The target of the SQL/DSF SYNC function can be another DB2/VSE database or any DB2 database that can be reached via DRDA, such as DB2/UDB or DB2 for z/OS.

1.1.1 Data Capturing

Applying data modifications from a source database to a target database implies that these changes have been captured in the source database.

The data capturing function is performed by the SQL/Auditing Facility, operating in synergy with the SQL/Monitoring Facility. These products are offered by Software Product Research. The Auditor saves all table changes to the **SQL/AF audit log** file as executable SQL statements. As far as the auditor is concerned, synchronization is a particular case of auditing.

Each table to be synchronized should be defined in the Auditor's **SQLAF RULES** file with the attributes **AUDIT CHANGE** and **COLUMN *** (which is the default).[2] If the table is also subject to read auditing, AUDIT ALL should be specified. This will also audit SELECT commands, which are ignored during SYNC processing. All SQL statements, both dynamic and compiled, are intercepted regardless of their origin, that is, data capturing is also performed for accesses issued by non-DB2/VSE users against DB2/VSE databases.

The SQL statements captured are written to the disk-resident SQL/AF audit log and archived to tape by the SQL/AF Archive Processor. The SQL/AF archive tapes can be kept as long as required.

[2]Defining specific column-names as trigger for sync auditing is possible: table synchronization would then occur only when the named columns are changed.

1.1.2 Performing Synchronization

A DB2/VSE table is synchronized with its target table in the target database by executing the **SQLDSF SYNC** command in the VSE environment.

The input file for the SYNC command is the **audit log** or an audit archive tape produced by the SQL/Auditing Facility. SYNC reads all audit entries for the designated table from the audit input file, extracts the UPDATE, DELETE, INSERT, ROLLBACK and COMMIT commands and executes them against the database specified in the SQLDSF **CONNECT TARGET** command.

Rolled-back LUW's are not applied to the target. COMMIT commands are executed in the same sequence as performed in the source database. The SYNC function does not perform COMMIT on its own initiative, except for SQL errors occurring during SYNC, which result in a ROLLBACK. Although stored audit entries may have been originated by multiple concurrently active LUW's, SYNC will never process more than one LUW at a time. Therefore, only one DB2/VSE agent structure is needed in the target database during synchronization.

If the target database is a non-DB2/VSE database, the program relies on the DRDA capabilities of DB2/VSE to execute the commands in the target database.

Note

Synchronization is performed in 2 steps:

- The first step selects from the SQL/AF audit log all log records that will be used for synchronization. The selected records are written to the SQLDSF WORKFILE. SQL/DSF acquires a SHARE LOCK on the table being synchronized. This ensures that no users are modifying the table. Read-only access to the table however is possible.

- The second step applies the log records from the workfile on the target database, except for those records which were found rolled-back during the first step.

1.1.3 Synchronization log

All SQL statements applied to the target database can be noted into a file called **<target_database> DSFLOG**. Rolled back statements also appear in the log, but their LUWid is enclosed in parentheses.

To enable logging, the **LOG ON** option should be specified in the **SQLDSF OPTIONS** member. The log can be printed or erased using the **PRINTLOG** or **ERASELOG** functions of SQLDSF.

1.1.4 Synchronization Mode

The SQL/AF log entries contain the expanded text of all SQL statements with all variables in the command replaced by the variable contents.[3]

For example: UPDATE EMPLOYEES SET DEPTNO = 5 WHERE EMPNO = 100

In most cases synchronization consists in executing this command against the target table in the target database. However, when the table row contains **LONG VARCHAR** columns, source table columns will be retrieved in view of synchronization. The pre-sync SELECT executed in these cases, should return **one table row** only. If the source table has a unique index, SQL/DSF will automatically use the related indexing columns in the SELECT WHERE clause. If no unique index exists, synchronization will use the source command predicates in such a way that a unique row is returned during SELECT. A logical table key can also be defined in the **SQLDSF RULES** file. For a description of the RULES syntax, see page 41.

1.1.5 Selective Synchronization

- A number of SQL/DSF statements allow to selectively synchronize a specified set of SQL/AF log entries, based on the DB2/VSE LUWID, the date or the time in the log entries, and so on.

- A **SUBSET** command provides for synchronization of those SQL/AF log entries that satisfy the defined SUBSET clause in the source table. This command is useful to synchronize **global** tables into tables containing a **subset** of the global table, for example, departmental tables.

[3]When commands access audited tables by means of **views**, the Auditor expands the view and ensures that the stored SQL statement contains real table names and real table column names only. Table or column names defined to SQL/DSF are always real names. They will automatically apply to eventual view names.

1.1.6 Scheduling synchronization

Explicit Scheduling

Synchronization is initiated by submitting the SQL/DSF SYNC command from a VSE jobstream. Input for the command is either the SQL/AF audit log on disk or an SQL/AF archive tape. In the latter case, an EXTRACT statement must be precede the SYNC statements.

Implicit Scheduling

Automatic synchronization can be performed using the SQL/DSF Scheduler (see page 39).

Scheduling Controls

To allow for multiple synchronizations to the same table during a DB2/VSE session, the SYNC command maintains the timestamp (the *"sync-stamp")* of the last committed synchronization LUW executed for a given table in a given target database in the SQLDSF_CONTROL_S table.

The sync-stamp is also used for restart after an SQL error on the target during synchronization. When the user has cleared the error situation on the target, restarting synchronization will automatically resume at the LUW that caused the error (and was rolled back).

Concurrent sync of the same table is not allowed and will result in an error message. Moreover, when a synchronization run did terminate abruptly (e.g. by an abend), that run **must** be restarted by the same VSE userid that started the run. Failure to do so will result in error message **SQLDSF907**.

1.1.7 Performance Considerations

Since SQL/DSF uses the **captured command** and not the DB2/VSE log data for synchronization, a single UPDATE or DELETE command that alters multiple rows, will be synchronized using a single transaction.

1.1.8 Using SQL/DSF for table restore

Since the audit log and archives contain, in executable format, all SQL statements that altered the table, the archives can be considered as incremental backup files. Consequently, the synchronization function can be used to incrementally restore a DB2 table using an audit log or audit archive. Moreover, a number of statements (FROM_DATE, TO_DATE, FROM_TIME, TO_TIME) allow to define the scope of synchronization, thus providing support for **point in time recovery**.

1.1.9 Restrictions

SQL/DSF does not perform specific processing for tables participating in a referential constraint. It is assumed that both source and target databases have an identical constraint definition, so that SQL statements executed against the target, also update the dependent tables by the target database server.

1.2 Table Transfer

The SQL/DSF table transfer function copies all rows of a designated table to the same table in a named target database. Like the synchronization function, table transfer is performed using a VSE jobstream or by the SQL/DSF Scheduler.

The **CONNECT SOURCE** statement identifies the database containing the table to be transferred. The **CONNECT TARGET** command designates the database that contains the target table. As for synchronization, SQL/DSF depends on the DRDA capabilities of DB2/VSE to connect a non-DB2/VSE database.

SQL/DSF transfers **data only**: the target table must have been created previously in the target database. By default, a transfer deletes (with an SQL DELETE) all rows in the target at the start of the transfer. If this is not desired, the NOPURGE option of the TRANSFER command must be specified.

Concurrent workunits

SQL/DSF maintains two LUW's during the table transfer. One LUW is used to SELECT the rows in the source database, the other to INSERT the selected rows in the target. As a result, SQL/DSF does **not** require intermediate disk or tape storage during the transfer operation.

Selective transfer

A **subset** of the source table can be transferred to the target by specifying a **SUBSET** command. Only those source table rows satisfying the SUBSET predicate are inserted in the target.

Index creation during transfer

When the transfer target is a DB2/VSE database, all table indexes will automatically be dropped before transfer and recreated when transfer completes. This results in better performance than upgrading the index(es) during transfer.

1.3 Table Compare

The SQL/DSF table compare function compares all rows of a named table with the mirror table in the designated target database. Like the transfer function, table compare is performed from a VSE jobstream.

The **CONNECT SOURCE** statement identifies the database containing the table to be compared. The **CONNECT TARGET** command designates the database that contains the target table. As for transfer, SQL/DSF depends on the DRDA capabilities of DB2/VSE to connect a non-DB2/VSE database.

The function compares the number of rows in both tables and the contents of each row. The compare mismatches if the number of rows is not identical, or if a difference is found in the row contents. In the latter case, the hexadecimal contents of the first mismatching rows are printed.

Contents comparing depends on the ORDER clause in the executed table SELECT, because the rows of both tables must be compared in the same sequence. SQL/DSF will attempt to determine the table order from a unique DB2/VSE index or from an SQL/DSF RULES KEY statement, if present. If no such index is found or if both tables are in a DRDA database, the first table column is assumed as the unique key column.

Selective compare

Specifying the **SUBSET** command during compare, will compare a **subset** of the source and target tables.

Concurrent workunits

SQL/DSF maintains two LUW's during table compare. One LUW is used to SELECT the rows in the source database, the other to SELECT in the target. As a result, SQL/DSF does **not** require intermediate disk or tape storage during the compare operation.

1.4 Conditional Table Transfer

The conditional transfer function is a combination of the compare and the transfer functions. A conditional transfer compares both tables and initiates a transfer when the compare fails.

The **CONNECT SOURCE** statement identifies the database containing the table to be compared. The **CONNECT TARGET** command designates the database that contains the target table.

The function compares the number of rows in both tables and the contents of each row. The compare mismatches if the number of rows is not identical, or if a difference is found in the row contents.

Selective transfer

A **subset** of the source table can be compared and transferred to the target by specifying the **SUBSET** command. Only the source table rows satisfying the SUBSET predicate will be compared and eventually transferred.

1.5 The SQL/DSF Scheduler

SQL/DSF provides the **SQLDSFS** Scheduler program to assist an installation in setting up a **DataSync server** environment. The Scheduler includes support for scheduling table synchronizations or transfers automatically and **chronologically**, as recorded in the **SQLDSFS RULES** file

Using the scheduler as an alternative for batch SQL/DSF commands, simplifies the management of the authorities necessary for table synchronization and transfer, since these can be granted to a single userid.

The SQLDSFS function is described in detail on page 39 of this manual.

1.6 Software Prerequisites

- VSE/ESA Version 1 Release 1 or later.
- DB2/VSE Version 6 Release 1 or later.
- SQL/Monitoring Facility Version 1 Release 1 or later. SQL/Monitoring Facility is a program product available from Software Product Research.
- SQL/Auditing Facility Version 1 Release 1 or later. SQL/Auditing Facility is a program product available from Software Product Research.

2 Installing the Data Synchronization Facility

2.1 Plan the DBspace needed for SQL/DSF

Each table synchronized by SQL/DSF to a target database requires a 34-character row in the SQL/DSF control table. Therefore the space needed for the SQL/DSF Dbspace depends on the number of “source” databases, the number of synchronized tables in those databases and the number of “target” databases into which these tables are synchronized.

During Dbspace ACQUIRE, the IBM defaults are used for PCTINDEX and PCTFREE. A Dbspace of 2048 pages should be sufficient in most cases.

2.2 Installing SQL/DSF

The SQL/DSF software is delivered as a PC file in ZIP format.

Place the ZIP file in a dedicated directory (e.g. SQLDSF) and unzip the file. Following files should now be present in the SQLDSF directory:

INSTALL.BAT	installation procedure for Windows
SEND2RDR.BAT	installation procedure for Windows
SQLDSF.PCF	SQL/DSF software
SQLDSSCF.PCF	software key
SQLDSFI1.VSEJOB	linkedit SQL/DSF
SQLDSFI2.VSEJOB	install SQL/DSF in the SQL/MF database
SQLDSFI3.VSEJOB	install SQL/DSF in a database
SQLDSF.BKS	bookshelf for IBM Library Reader
SQLDSF.BOO	SQL/DSF User’s Guide in IBM Library Reader format
SQLDSF.PDF	SQL/DSF User’s Guide in PDF format

2.2.1 Preliminary Note

The INSTALL.BAT and SEND2RDR.BAT installation procedures use the “SEND.EXE” to upload the PC files to the POWER reader queue. Ensure that your 3270 emulator supports SEND (some emulators don’t) and that the EXE is on your active path.

If you cannot use the SEND.EXE, use the upload facilities of your emulator to perform the equivalent of the SEND commands contained in the BAT files, that is:

for the INSTALL.BAT:

```
SEND SQLAFSCF.PCF (FILE=RDR BINARY LRECL=80 NOUC
SEND SQLAF.PCF (FILE=RDR BINARY LRECL=80 NOUC
```

for the SEND2RDR.BAT:

```
SEND <filename> (FILE=RDR
```

2.2.2 Upload the SQL/DSF software

- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Execute the INSTALL.BAT by clicking. This will upload 2 jobs to the POWER/VSE reader queue with DISP D and class 0. The jobs catalog all SQL/DSF components into the VSE library previously chosen as residence for the prerequisite product SQL/MF
- After job completion, check the DISP=H listing for any errors.

2.2.3 Link SQL/DSF

- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Drag and drop the file SQLDSFI1.VSEJOB on the SEND2RDR.BAT. This will send and start the job in class 0 and DISP D. The output listing is in DISP H.
- Check the output listing for errors.

2.2.4 Install SQL/DSF in the SQL/MF database

This installation step should be executed in the database used as the SQL/MF “LOG” database. This step will load the SQL/DSF packages into the database, define the SQLDSF userid to DB2 with DBA authority and create the SQL/DSF synchronization control tables.

Edit the PC file SQLDSFI2.VSEJOB and assign the following SETPARM symbols

CAT
The VSAM catalog for a SAM ESDS workfile (default is VSESPUC)
DB
The name of the SQL/MF LOG database.
DBAPASS
The password of user SQLDBA in that database.
DSFPASS
The password for the new user SQLDSF in that database.
POOL
The DB2 storage pool where the SQL/DSF tables will reside.
PAGES
The number of pages for the SQL/DSF Dbspace.

Submit the SQLDSFI2.VSEJOB as follows:

- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Drag and drop the file SQLDSFI2.VSEJOB on the SEND2RDR.BAT. This will send and start the job in class 0 and DISP D. The output listing is in DISP H.
- Check the output listing for errors.

2.2.5 Install SQL/DSF in a database

This installation step should be executed in all databases where SQL/DSF will be used. This step will load the SQL/DSF packages into the database and define the SQLDSF userid to DB2 with DBA authority.

Edit the PC file SQLDSFI3.VSEJOB and assign the following SETPARM symbols

CAT
The VSAM catalog for a SAM ESDS workfile (default is VSESPUC)
DB
The name of the target database.
DBAPASS
The password of user SQLDBA in that database.
DSFPASS
The password for the new user SQLDSF in that database.

Submit the SQLDSFI3.VSEJOB as follows:

- Start ICCF PC file transfer (fast path 386) in 3270 emulator session A.
- Drag and drop the file SQLDSFI3.VSEJOB on the SEND2RDR.BAT. This will send and start the job in class 0 and DISP D. The output listing is in DISP H.
- Check the output listing for errors.

3 Invoking SQL/DSF functions

- Using VSE JCL.
- By inserting the function request in the **SQLDSFS RULES** file of the SQL/DSF Scheduler.[4]

SQL/DSF statements are divided into **declarative** statements and **executive** statements. The executive statements are **SYNC, TRANSFER, CTRANSFER, EXTRACT** and **COMPARE**. All other statements are declarative.

The DB2 userid specified in the CONNECT statement should have DBA authority in both the source and in the target database. If the CONNECT does not specify a userid, the SQL/DSF function will be executed under userid SQLDSF, which has been defined with DBA authority during SQL/DSF installation.

[4]The Scheduler is a VSE partition that runs the **SQLDSFS** program, which is part of the program product.

4 Performing table synchronization

Prerequisites

The table to be synchronized must have been defined previously in the SQLAF RULES file of the SQL/AF Auditing Facility with an audit option ensuring that inserts, deletes and updates are captured (for example, AUDIT ALL or AUDIT CHANGE).

The target table must be **compatible** with the source table as follows:

- all columns defined in the source table must be defined in the target table
- all columns defined in the target, but not in the source table, should be nullable
- all column names must be identical
- all column datatypes must be identical or compatible[5]

Operation

To perform a table synchronization, following SQL/DSF commands are always required:

- a **CONNECT SOURCE** statement designating the name of the source database
- a **CONNECT TARGET** statement designating the name of the target database
- a **SYNC** statement naming the table to be synchronized

If SYNC must handle LONG VARCHAR columns and the source table has no unique index, the SQLDSF RULES KEY command may be required as well.

Invocation

Invoke the SQLDSF program to perform synchronization, as shown below:

```
// JOB xxxx
// DLBL DSFWORK,'%DSWRK',0,VSAM,CAT=xxxxxxxx,RECSIZE=9000,          +
        DISP=(NEW,DELETE),RECORDS=(1000,100)
// EXEC SQLDSF,SIZE=SQLDSF
CONNECT TO SOURCE xxxxxxxx
CONNECT TO TARGET xxxxxxxx
SYNC creator.tablename
/*
/&
```

Notes

- A DLBL for the SQLDSF SAM workfile is required. As in the above example, the workfile can be in VSAM managed space.
- SQLDSF executes in both static and dynamic partitions.

[5] A datatype is compatible when the source column can be moved to the corresponding target column. For example: a SMALLINT source column is compatible with a column defined as INTEGER in the target.

4.1 Declarative SYNC statements

Declarative statements are specified before the SYNC statement. If multiple SYNC statements occur in the jobstream, the declaratives need be specified only once. However, the declarative **SUBSET** is cleared at the end of each SYNC and must be respecified, if needed.

4.1.1 CONNECT SOURCE

Syntax: **CONNECT [user IDENTIFIED BY password] TO SOURCE database**

Specify the name of the DB2/VSE database that is the source for synchronization. If a userid is specified, this userid should have DBA authority in the source database. If no userid is specified, connect is done under userid SQLDSF, which was defined during installation as DBA .

Synchronization uses the source connection mainly to obtain DB2/VSE catalog information about the table being synchronized. If synchronization involves processing of LONG VARCHAR columns, the connection is also used to obtain the data values for such columns.

4.1.2 CONNECT TARGET

Syntax: **CONNECT [user IDENTIFIED BY password] TO TARGET database**

Specify the name of the database that is the target for SYNC. If a userid is specified, this userid should have DBA authority in the target database. If no userid is specified, connect is done under userid SQLDSF, which was defined during installation as DBA.

4.1.3 FROM_LUW

Syntax: **FROM_LUW n**

The statement allows to start synchronization from a specified SQL LUWID. You can display the LUWID for specific log entries, using the **Log Scan** program, which is described in the SQL/AF User's Guide.

4.1.4 TO_LUW

Syntax: **TO_LUW n**

Used in combination with FROM_LUW, the statement performs synchronization for a range of SQL LUWIDs, that is, from FROM_LUW up to and including TO_LUW.

4.1.5 FROM_DATE

Syntax: **FROM_DATE expression**

The statement selects SQL/AF audit entries for a specified date. Any valid SQL date expression can be used such as: CURRENT DATE, CURRENT DATE - 1 DAY etc.

4.1.6 TO_DATE

Syntax: **TO_DATE expression**

Used in combination with FROM_DATE, the statement performs synchronization for the specified date range, that is, from FROM_DATE up to and including TO_DATE.

4.1.7 FROM_TIME

Syntax: **FROM_TIME expression**

The statement selects SQL/AF audit entries greater than or equal to a specified time. Any valid SQL time expression can be used, such as: CURRENT TIME, CURRENT TIME - 30 MINUTES, 08.30.00 etc.

4.1.8 TO_TIME

Syntax: **TO_TIME expression**

Used in combination with FROM_TIME, the statement performs synchronization for the specified time range, that is, from FROM_TIME up to and including TO_TIME.

NOTE:
If more than one of the above selection statements are used, the log entries must satisfy **all** specified conditions before being selected.

4.1.9 SUBSET

Syntax: **SUBSET expression**

This command designates that SYNC is for a **subset** of the source table and identifies the subset in the expression.

A SUBSET expression has the following syntax:

<column-name> **<operator>** **<constant>**

where:
<operator> is one of the logical operators = ^= <> > < >= <=
<constant> is a string or a date expression

Coding rules

(1) Enclosing quotes are not required for alphabetic constants, unless the constant contains embedded blanks.
(2) With the exception of date expressions, no column or scalar functions are allowed in “constant”.
(3) Generically testing the leading positions of a column is requested by terminating the constant with a % sign.
(4) When testing numerical constants, the following rules should be observed:
- leading zeroes should **not** be specified
- trailing fractional zeroes **must** be specified
- negative values should have a leading **-** sign
- a decimal point is entered with a **.** sign

(5) Multiple expressions may be connected through AND / OR. These connectors should not be combined, that is, the SUBSET clause should contain only AND or OR connectors.
(6) Composite expressions must be flat: they cannot contain parentheses.

SQL/DSF evaluates the SUBSET expression as follows:
- SQL/DSF applies the SUBSET clause on INSERT commands only: synchronized DELETE and UPDATE commands are assumed to supply the SUBSET selection in their WHERE clause.
- The INSERT command is scanned for the occurrence of the SUBSET column-name in the INSERT column-list and for the occurrence in the corresponding position of the INSERT VALUES list, of a value satisfying the SUBSET condition.

The only purpose of the SUBSET command is to define a logical subset within a global table. The command should **not** be used to apply the same log entries to the same table in the same database in multiple SYNC runs. This is not possible, as a SYNC run always resumes after the last recorded sync-stamp for a given table in a given database

SUBSET command samples

```
SUBSET DEPARTMENT=10
SUBSET DEPARTMENT IN (10,11)
SUBSET INVOICE_DATE = CURRENT_DATE - 1 DAY
```

4.2 Executive SYNC statement

The **SYNC** statement extracts the DELETE, INSERT, UPDATE, COMMIT and ROLLBACK commands related to designated tables from the SQL/AF log and applies them to the target table in the target database.

Syntax: **SYNC creator.tablename [OPTIONS ...]**

Performs synchronization for the named source table.

4.2.1 SYNC OPTIONS clause

The OPTIONS clause on the SYNC statement has following syntax:

OPTIONS [VERBOSE] [EXIT exitname [exit_arguments]]

- The **VERBOSE** option is provided for tracing purposes. It displays additional information on SYSLOG during sync.

- The **EXIT** option specifies the name of a statement user exit. The exit is coded as a REXX exec and is invoked for each SQL statement that will be applied to the target database during sync. The exit can modify the statement text.

  Up to 16 arguments can be passed to the exit. The length of each argument should not exceed 8 characters and should not contain blanks.

  If specified, the **EXIT** option should be the last option coded on the SYNC statement.

  How to code a statement user exit is described on page 45.

4.2.2 SYNC Sample

```
CONNECT TO SOURCE DBNA
CONNECT TO TARGET DBNB
SYNC SQLDBA.CUSTOMER
```

5 Using the Extract function

Operation

Although the SQL/AF audit log is usually the input for table synchronization, synchronization can also be done using an SQL/AF archive tape. The EXTRACT function copies the SQL/AF archive tape defined by the **TLBL IN** to the sequential disk file XTRWORK. All SYNC commands that follow the EXTRACT command will use XTRWORK as input, instead of the SQL/AF audit log.

All tables to be synchronized using the specified archive tape must be processed within the same VSE job.

Invocation

```
// JOB xxxx
// ASSGN SYS020,cuu
// MTC REW,cuu
// TLBL IN,'SQLAF_archivename',,volid,,,,,DISP=OLD
// DLBL DSFWORK,'%DSWRK',0,VSAM,CAT=xxxxxx,RECSIZE=9000,          +
          DISP=(NEW,DELETE),RECORDS=(1000,100)
// DLBL XTRWORK,'%XTWRK',0,VSAM,CAT=xxxxxx,RECSIZE=9000,          +
          DISP=(NEW,DELETE),RECORDS=(1000,100)
// EXEC SQLDSF
EXTRACT
CONNECT TO SOURCE xxxxxxxx
CONNECT TO TARGET xxxxxxxx
SYNC creator.tablename_1
.
.
SYNC creator.tablename_n
/*
/&
```

6 Transferring Tables

Prerequisites

The table to be transferred must be compatible with the receiving table in the target database, as follows:

- the number of columns must be the same in both tables
- the column order must be identical
- the column datatypes must be identical or compatible
- the target table column names are not significant, as the INSERT issued to the target during transfer does not specify a column-list

Invocation

The TRANSFER function is controlled by the statements CONNECT SOURCE, CONNECT TARGET and TRANSFER. These statements are mandatory. An optional SUBSET command may be used to transfer a specified set of table rows.

Invoke the SQLDSFU program to transfer a table, as shown below:

```
// JOB xxxx
// EXEC SQLDSFU,SIZE=SQLDSFU
CONNECT TO SOURCE xxxxxxxx
CONNECT TO TARGET xxxxxxxx
TRANSFER creator.tablename
/*
/&
```

6.1 CONNECT SOURCE

Syntax: **CONNECT [user IDENTIFIED BY password] TO SOURCE database**

Specify the name of the DB2/VSE database that is the source for transfer. If no userid is specified, connect is done under userid SQLDSF, which was defined during installation as DBA.

Transfer uses the source connection to obtain DB2/VSE catalog information about the table being transferred.

6.2 CONNECT TARGET

Syntax: **CONNECT [user IDENTIFIED BY password] TO TARGET database**

Specify the name of the database that is the target for transfer. If a userid is specified, this userid should have DBA authority in the target database. If no userid is specified, connect is done under userid SQLDSF, which was defined during installation as DBA

6.3 SUBSET expression

This command is used to transfer a defined set of table rows to the target. The SUBSET expression can be any SQL expression, valid in the SELECT WHERE clause for the source table.

6.4 TRANSFER

TRANSFER creator.tablename
[NOPURGE]
[DELETE_ROWS]
[NOINDEX]
[COMMITCOUNT {100000 | n}]

TRANSFER copies the named table from the source database to an identically named table in the target database.

Command Options

- Before starting transfer, the target table is purged by issuing an unqualified DELETE (if SUBSET omitted) or a DELETE WHERE <SUBSET> specification. Specify the **NOPURGE** option to prevent this initial DELETE. Transfer then will **append** the target rows to the table. For large tables, it is preferable to specify NOPURGE and to clear the target table by re-creating it, prior to transfer. This will improve performance and avoid empty table pages, resulting from the DELETE.
- The DELETE that implements the pre-transfer purge may eventually fill the DB2 log (SQLCODE -933) when transferring a large table. The **DELETE_ROWS** option will prevent this, by performing a cursored DELETE with a COMMIT at regular intervals.
- The **NOINDEX** option avoids the DROP / CREATE INDEX commands that are issued by default before / after a transfer. The option is useful when transferring a small subset to a large table.
- To avoid log full conditions, the transfer function performs a COMMIT after transferring <**COMMITCOUNT**> rows to the target. The default COMMITCOUNT is 100000 rows.

Notes

- The TRANSFER function does not require disk or tape storage for its operation. It uses **VSE subtasks** to maintain a concurrent connection to the source and to the target database.
- If SQL/Auditing has been enabled for the table in the target database, data access resulting from TRANSFER is not audited.
- When transferring a subset of a table, the SUBSET clause is also used to DELETE the target rows before transfer. Omitting the SUBSET will result in a DELETE of the entire target table.

6.5 Duplicate key handling during subset transfer

When an SQLCODE -803 occurs during transfer of a table subset, the offending row is automatically deleted on the target and the insert is retried.

To activate this facility, the SQLDSF RULES file should specify a key for the source table being transferred. This key is used by SQLDSF to perform a DELETE WHERE. The column names specified in the KEY statement should allow for the selection of the row to be deleted.

Since detection of SQLCODE -803 depends on the existence of a unique index, the TRANSFER NOINDEX option is automatically forced during a subset transfer.

6.6 Sample

```
CONNECT TO SOURCE DBNA
CONNECT TO TARGET DBNB
TRANSFER SQLDBA.TAB1 COMMITCOUNT 10000
```

6.7 Transfer performance considerations

- the transfer operation performs a COMMIT after a defined number of rows have been transferred
- before transfer, all table indexes are dropped; these indexes are recreated when transfer is complete
- the best transfer performance is obtained by the following sequence of operations:
    - drop the target table using a drop DBspace
    - re-create the target table and its dependent objects
    - perform a TRANSFER NOPURGE

7 Comparing Tables

Prerequisites

The column definitions of the tables to be compared must be compatible as follows:

- the number of columns must be the same in both tables
- the column order must be identical
- the column datatypes must be identical
- the column names are not significant

Invocation

The function is controlled by the statements CONNECT SOURCE, CONNECT TARGET and COMPARE. An SQL/DSF RULES KEY statement is required when SQL/DSF is unable to determine table sequencing automatically, because a unique index is not available. An optional SUBSET command may be used to compare a specified set of table rows.

Invoke the SQLDSFU program to compare a table, as shown below:

```
// JOB xxxx
// EXEC SQLDSFU,SIZE=SQLDSFU
CONNECT TO SOURCE xxxxxxxx
CONNECT TO TARGET xxxxxxxx
COMPARE creator.tablename
/*
/&
```

7.1 CONNECT SOURCE

Syntax: **CONNECT [user IDENTIFIED BY password] TO SOURCE database**

Specify the name of the DB2/VSE database that is the source for compare. If no userid is specified, connect is done under userid SQLDSF, which was defined during installation as DBA. Compare uses the source connection mainly to obtain DB2/VSE catalog information about the table being compared.

7.2 CONNECT TARGET

Syntax: **CONNECT [user IDENTIFIED BY password] TO TARGET database**

Specify the name of the database that is the target for transfer. If a userid is specified, this userid should have DBA authority in the target database. If no userid is specified, connect is done under userid SQLDSF, which was defined during installation as DBA

7.3 SUBSET expression

This command is used to compare a defined set of table rows. The SUBSET expression can be any SQL expression valid in the WHERE clause of the SELECT for the source and target tables.

7.4 COMPARE

COMPARE creator.tablename

The command compares the named table in the source database to an identically named table in the target database.

On exit from the compare command, the VSE returncode is zero if the tables match and **700** if they do not.

The COMPARE function does not require disk or tape storage for its operation. It uses **VSE subtasks** to maintain a concurrent connection to the source and to the target database.

7.5 Sample

Compare a table in two databases.

```
CONNECT TO SOURCE DBNA
CONNECT TO TARGET DBNB
COMPARE SQLDBA.TAB1
```

8 Conditional Table Transfer

Prerequisites

The column definitions of the tables to be compared must be compatible as follows:

- the number of columns must be the same in both tables
- the column order must be identical
- the column datatypes must be identical
- the column names are not significant

Invocation

The CTRANSFER function is controlled by the statements CONNECT SOURCE, CONNECT TARGET and CTRANSFER. These statements are mandatory. An optional SUBSET command may be used to transfer a specified set of table rows.

Invoke the SQLDSFU program to transfer a table, as shown below:

```
// JOB xxxx
// EXEC SQLDSFU,SIZE=SQLDSFU
CONNECT TO SOURCE xxxxxxxx
CONNECT TO TARGET xxxxxxxx
CTRANSFER creator.tablename
/*
/&
```

8.1 CONNECT SOURCE

Syntax: **CONNECT [user IDENTIFIED BY password] TO SOURCE database**

Specify the name of the DB2/VSE database that is the source for transfer. If no userid is specified, connect is done under userid SQLDSF, which was defined during installation as DBA. Transfer uses the source connection to obtain DB2/VSE catalog information about the table being transferred.

8.2 CONNECT TARGET

Syntax: **CONNECT [user IDENTIFIED BY password] TO TARGET database**

Specify the name of the database that is the target for transfer. If a userid is specified, this userid should have DBA authority in the target database. If no userid is specified, connect is done under userid SQLDSF, which was defined during installation as DBA

8.3 SUBSET expression

This command is used to compare - and eventually to transfer - a defined set of table rows. The SUBSET expression can be any SQL expression valid in the WHERE clause of the SELECT for the source and target tables.

8.4 CTRANSFER

CTRANSFER creator.tablename

The command compares the named table in the source database to an identically named table in the target database. If the compare is not successful, because there is a difference in number-of-rows or row contents, a table transfer is initiated. Before starting transfer, the target table is purged by issuing an unqualified DELETE (if SUBSET omitted) or a DELETE WHERE <SUBSET> specification.

The CTRANSFER function does not require disk or tape storage for its operation. It uses **VSE subtasks** to maintain a concurrent connection to the source and to the target database.

8.5 Sample

```
CONNECT TO SOURCE DBNA
CONNECT TO TARGET DBNB
CTRANSFER SQLDBA.ACCOUNTS
```

Compares the ACCOUNTS table in databases DBNA and DBNB and transfers it if mismatching.

9 Using the SQL/DSF Scheduler

The SQL/DSF scheduler program **SQLDSFS** runs in a VSE partition. It is started by submitting the following JCL:

```
// JOB SQLDSFS
// EXEC SSPRLIB
// LIBDEF PROC,SEARCH=&SPRLIB
// EXEC SQLDSFS,SIZE=SQLDSFS
/*
/&
```

SQLDSFS may be requested to perform a defined command stream at a defined time of day. These specifications are recorded in a control file named **SQLDSFS RULES** which is a member in the VSE library SPRLIB.

Each group of commands to be scheduled must be preceded by a TIME or EVERY command.

TIME hh:mm
Specify the time of execution for the commands that follow.

EVERY n [FROM hh:mm TO hh:mm]
Specify the interval in minutes for the execution of the commands that follow. If the FROM / TO clause is specified, execution is performed during the defined period only.

When the specified time has been reached, SQLDSFS processes all commands following the TIME or EVERY command until a new TIME or EVERY command is encountered or until end-of-file. When end-of-file is found, the Scheduler pauses until midnight and then restarts the RULES from begin-of-file.

Following commands can appear in the RULES:

- a valid SQL/DSF command such as CONNECT, SYNC, TRANSFER etc.
- comment lines prefixed with “*”

Notes

- Before processing them, the entries in the RULES file are chronologically ordered by SQLDSFS.
- When an error occurs during processing of a command, the remaining commands for the current logical function are flushed and processing continues with the next logical function, if any. For instance, if multiple SYNCs are requested at a given TIME and one of them fails, the other SYNC commands will be processed.
- When SQLDSFS is restarted manually, the first RULES entry processed will be the one with a TIME statement greater than the restart time.
- If the RULES file is modified later on and if the changes should have **immediate** effect, the scheduler must be restarted. To perform an orderly restart after completion of any outstanding work, issue a VSE **MSG** command to the partition.

Sample SQLDSFS RULES

```
EVERY 5 FROM 08:00 TO 18:00
CONNECT TO SOURCE DBN1
CONNECT TO TARGET DBN2
SYNC SQLDBA.Q_REQUESTS
*
TIME 20:00
CONNECT TO SOURCE DBN1
CONNECT TO TARGET DBN2
SYNC SQLDBA.CUSTOMERS
SYNC SQLDBA.INVOICES
TRANSFER SQLDBA.INV_CODES
```

Contents of sample rules file:

- From 8 a.m. until 6 p.m. synchronize the Q_REQUESTS table every 5 minutes.
- At 20h. synchronize the customer table and invoice tables to DBN2, using the SQL/AF audit log and transfer the invoice code table.

10 SQL/DSF RULES file

The purpose of the RULES file is to supply the logical key for tables without a unique index. This specification may be needed during SYNC[6], COMPARE or CTRANSFER commands. The specification is also used to provide for automatic delete of duplicate keys during a subset transfer.

The **SQLDSF RULES** file should be stored as a member in the VSE library SPRLIB.

A logical definition consists of a **DATABASE**, a **TABLE,** and a **KEY** command. The syntax of these commands is as follows:

DATABASE nnnnnnnn

Specify the name of the source database containing the table.

TABLE creator.table-name

Specify the creator and tablename for which the logical key is being defined.

KEY column-name [,column-name-2, ... column-name-n]

Specify one or more column-names that make up the logical table key. Multiple column-names must be separated by a comma. The column-names specified are not necessarily indexing columns.

[6]Only when commands processing LONG VARCHAR columns are synchronized.

11 SQL/DSF OPTIONS file

The SQLDSF OPTIONS file may specify following execution options:

VERBOSE

The **VERBOSE** option is provided for tracing purposes. It displays additional information on SYSLOG during sync.

LOG ON | OFF

The **LOG ON** option requests that all SQL statements applied to the target database during synchronization will be noted into a file called **<target_database> DSFLOG**. Rolled back statements also appear in the log, but their LUWid is enclosed in parentheses.

12 SQL/DSF User Exit

The statement user exit is invoked during table synchronization. The exit can inspect and modify each SQL statement executed during synchronization.

There is no predefined name for the statement user exit and an installation may have multiple exits. The SYNC statement names the exit to be invoked for a given synchronization run, as described on page 28.

When the exit is invoked, it receives following arguments:

- name of the source database
- name of the target database
- pointer to the SQL statement that will be applied
- length of the SQL statement
- if specified, the arguments from the SYNC EXIT statement

The exit uses the REXX **STORAGE** builtin function to retrieve the SQL statement and to modify it.

To facilitate word scanning of the SQL statement, it is passed in normalized format, that is, with a blank between each syntactical element.

For example:

the statement UPDATE TEST SET COL1='XXX' WHERE COL2='ZZZ'
is passed as UPDATE TEST SET COL1 = 'XXX' WHERE COL2 = 'ZZZ'

The exit should pass a returncode as follows:

- If the statement should not be synchronized to the target database, a negative value should be returned.
- If the statement has not been modified by the exit, a zero value should be returned.
- If the statement has been modified, the current length of the statement should be passed on exit.

The exit is called by SQLDSF as a separate VSE subtask, which can be used by the exit to execute SQL statements. No database connection exists when the exit is entered.

Sample Statement Exit

```
/*
     Sample synchronization exit


 Entry arguments :  source database
                    target database
                    pointer to SQL statement that will be synced
                    length of statement

 Returncode      :  < 0 if statement should not be synced
                    = 0 if statement must be synced without change
                    > 0 if a changed statement must be synced
                        with the length specified in returncode
*/

 address command

 arg source_db target_db statement_pointer statement_length

 return_value = 0         /* assume statement applied without change */

/* Get the SQL statement into the variable "statement" */

 statement=storage(statement_pointer,statement_length)

 if <statement should not be applied on target> then return_value=-1

 else

/*
    If needed, modify the "statement" variable.

    Modifying the statement text for SQL/DSF is done as follows:

    x=storage(statement_pointer,length(statement),statement)
    return_value=length(statement)

*/

 exit return_value
```

13 SQL/DSF Control Tables

Following DB2/VSE tables are maintained by SQL/DSF to control its operations:

13.1 SQLDSF_CONTROL_S

Column	Datatype	Contents
TCREATOR	CHAR(8)	creator of the synchronized table
TNAME	CHAR(18)	name of the synchronized table
DBN	CHAR(8)	name of sync target database
OWNER	CHAR(8)	userid performing SYNC - blank if no sync in progress
SYNCTOD	CHAR(8)	timestamp of last audit log record synchronized to target database (in hardware clock format)

Maintains synchronization control information for each source table processed in each target database. A row is inserted for every table synchronized on each target database.

13.2 SQLDSF_CONTROL_P

A table used internally to evaluate date expressions.

14 Messages

SQLDSF001 **Syntax error in the above statement.**

When a syntax error is detected, the statement in error is displayed and another error message may follow.

Action Correct the statement in error and resubmit the SQL/DSF command stream.

SQLDSF101 **SQLCODE validating date/time expression is *n***

An invalid date or time expression was entered on the commands FROM_DATE, TO_DATE, FROM_TIME or TO_TIME.

n is the SQLCODE received during expression evaluation

Action Correct the statement in error and resubmit the SQL/DSF command stream.

SQLDSF102 **SQLCODE connecting target database is *n***

n is the SQLCODE received when attempting to connect the target database during SYNC, COMPARE or TRANSFER.

Action Examine the SQLCODE. Ensure that the target database is active and that you have the privileges to connect it.

SQLDSF103 **An SQL error has occurred**

A negative SQLCODE has been received after an SQL statement during SYNC, COMPARE or TRANSFER. Messages SQLDSF104 to SQLDSF108 will further explain the error.

SQLDSF104 **SQLCODE *n***
SQLERRM *nnn*
SQLERRD1 *n*

The above SQLCA fields explain the SQL error signalled by message SQLDSF103.

Action Use the above data to determine the nature of the SQL error.

SQLDSF105 **SQLSTATE *n***

The above item explains the SQL error signalled by message SQLDSF103.

Contrarily to the SQLCODE, which slightly varies across IBM/DB2 platforms, the SQLSTATE value is unambiguous in all these systems.

SQLDSF106 LUWID *n*

When the SQL error occurs during a SYNC operation, the number of the originating LUW in the source database is shown.

SQLDSF107 SYNCSTAMP *n*

When the SQL error occurs during a SYNC operation, the message shows the SQL/AF stamp for the failing command. The syncstamp is identical for all commands in a given SQL LUW. It represents the hexadecimal value of the hardware clock at the begin of the LUW.

SQLDSF108 SQL Command Text:

When the SQL error occurs during a SYNC operation, the message shows the original command text causing the error.

SQLDSF110 SQLCODE connecting source database is *n*

n is the SQLCODE received when attempting to connect the source database during SYNC, COMPARE or TRANSFER.

Action Examine the SQLCODE. Ensure that the source database is active and that you have the privileges to connect it.

SQLDSF111 KEY command is required

During COMPARE or CTRANSFER, the KEY specification has been omitted and SQL/DSF was unable to retrieve a key definition from a unique index in SYSINDEXES. During SYNC, a unique index was not available and the pre-SYNC SELECT resulted in the retrieval of more than one table row.

Action A KEY command must be supplied in the SQLDSF RULES file.

SQLDSF112 SQLCODE connecting SQL/MF database is *n*

n is the SQLCODE received when attempting to connect the SQL/MF database during static SYNC.

Action Examine the SQLCODE. Ensure that the SQL/MF database is active and that you have the privileges to connect it.

SQLDSF113 SQLCODE selecting SQL/MF STMNTS table is *n*

n is the SQLCODE received when selecting the program command text from the SQL/MF SQLCS_SQL_STMNTS table during static SYNC.

Action Examine the SQLCODE. Ensure that the SQL/MF database is active and that you have the necessary SQL privileges..

SQLDSF114 SQLCODE selecting SQLDSF_CONTROL is *n*

n is the SQLCODE received when accessing the SQLDSF_CONTROL table during static SYNC.

Action Examine the SQLCODE. Ensure that the SQL/MF database is active and that you have the necessary SQL privileges to access the control table.

SQLDSF115 SQLCODE inserting SQLDSF_CONTROL is *n*

n is the SQLCODE received when accessing the SQLDSF_CONTROL table during static SYNC.

Action Examine the SQLCODE. Ensure that the SQL/MF database is active and that you have the necessary SQL privileges to access the control table.

SQLDSF116 Syntax error in SQLDSF RULES dataset

A syntax error was detected when reading the RULES dataset to obtain the table key during SYNC or COMPARE.

Action Correct the RULES statement in error.

SQLDSF201 File system error. Returncode is *n*

A VSAM error has occurred when reading an SQL/DSF input file. The returncode is the VSAM error code from the RPL.

SQLDSF701 Compare is in progress

Informatory message issued at the start of a COMPARE command.

SQLDSF702 *n* FROM rows compared

Informatory message showing the progress of the compare operation. The message is issued each time 100000 table rows have been compared.

SQLDSF703 *n* FROM rows compared

Informatory message issued at the end of compare. It shows the total number of rows compared on the source table.

SQLDSF704 Tables do compare

Informatory message indicating that both tables are identical.

SQLDSF705 Tables do not compare

Informatory message indicating that the compared tables are not identical, either because their rowcount does not match or because there is a difference in row contents. In the latter case, the hexadecimal contents of the first mismatching row are printed. This is not done during a CTRANSFER command, in which case the message triggers a table transfer.

Action Ensure that both tables are processed in the same sequence. If needed, supply a KEY command.

SQLDSF706 *n* rows in table *nnn*

If the tables do not match, the message is issued twice, showing the number of rows in both the source and the target table.

SQLDSF707 Number of mismatching rows is *nnn*

If the tables do not match, the message shows how many table rows have a different content.

SQLDSF801 Transfer is in progress

Informatory message issued at the start of a TRANSFER command.

SQLDSF802 *n* rows transferred

Informatory message showing the progress of the transfer operation. The message is issued each time 100000 table rows have been transferred.

SQLDSF896 *n* duplicate rows deleted on target

A duplicate key condition (SQLCODE -803) occurred *n* times during table transfer. Using the KEY specification in SQLDSF RULES, these duplicate rows were removed.

SQLDSF897 TRANSFER to target *nnn* successfully completed

Informatory message issued at the normal completion of a TRANSFER operation to database ***nnn***.

SQLDSF898 TRANSFER to target *nnn* abnormally completed

Informatory message issued at the abnormal completion of a TRANSFER operation to database ***nnn***. Another message will precede and explain the nature of the error.

SQLDSF899 *n* rows transferred

Informatory message issued at the completion of a TRANSFER operation. It shows the total number of rows transferred.

SQLDSF901 SYNC is in progress

Informatory message issued at the start of a SYNC command.

SQLDSF902 ***n*** **SYNC requests processed**

Informatory message showing the progress of the synchronization operation. The message is issued after processing 10000 SQL/AF log entries.

SQLDSF907 Synchronization process currently owned by VSE user *nnn*

A synchronization run for the table has already been started by the named VSE user. Concurrent synchronization is not allowed. If a previous synchronization run came to a non-controlled completion, the run must be restarted by the VSE user **nnn.**

SQLDSF914 Audit archive records pending for sync

The audit log was archived to tape and a request is made to synchronize from the current audit log. Synchronization must be performed using the audit archive before processing the disk log.

SQLDSF915 User invoking SQL/DSF SYNC must have DBA authority

Users invoking synchronization must be DBA. This is checked by SQL/DSF in the source database and in the target database, if the latter is a DB2/VSE server.

SQLDSF994 ***n*** **target rows deleted**

Informatory message issued at the completion of a SYNC operation. It shows the number of rows deleted from the target table due to synchronization.

SQLDSF995 ***n*** **target rows inserted**

Informatory message issued at the completion of a SYNC operation. It shows the number of rows inserted into the target table due to synchronization.

SQLDSF996 ***n*** **target rows updated**

Informatory message issued at the completion of a SYNC operation. It shows the number of rows updated on the target table due to synchronization.

SQLDSF997 SYNC to target *nnn* successfully completed

Informatory message issued at the normal completion of a SYNC operation to database ***nnn***.

SQLDSF998 SYNC to target *nnn* abnormally completed

Informatory message issued at the abnormal completion of a SYNC operation to database ***nnn***. Another message will precede and explain the nature of the abnormal termination.

SQLDSF999 Number of log records scanned: n

Informatory message indicating how many audit log records were read during the SYNC process.