Database

Administrator Module

NORSK DATA A.S

Database

Administrator Module

REVISION RECORD

Revision	Notes
10/77	Original Printing

Publ. No. ND-60.097.01
October 1977

NORSK DATA A.S.

Lørenveien 57, Postboks 163 Økern, Oslo 5, Norway

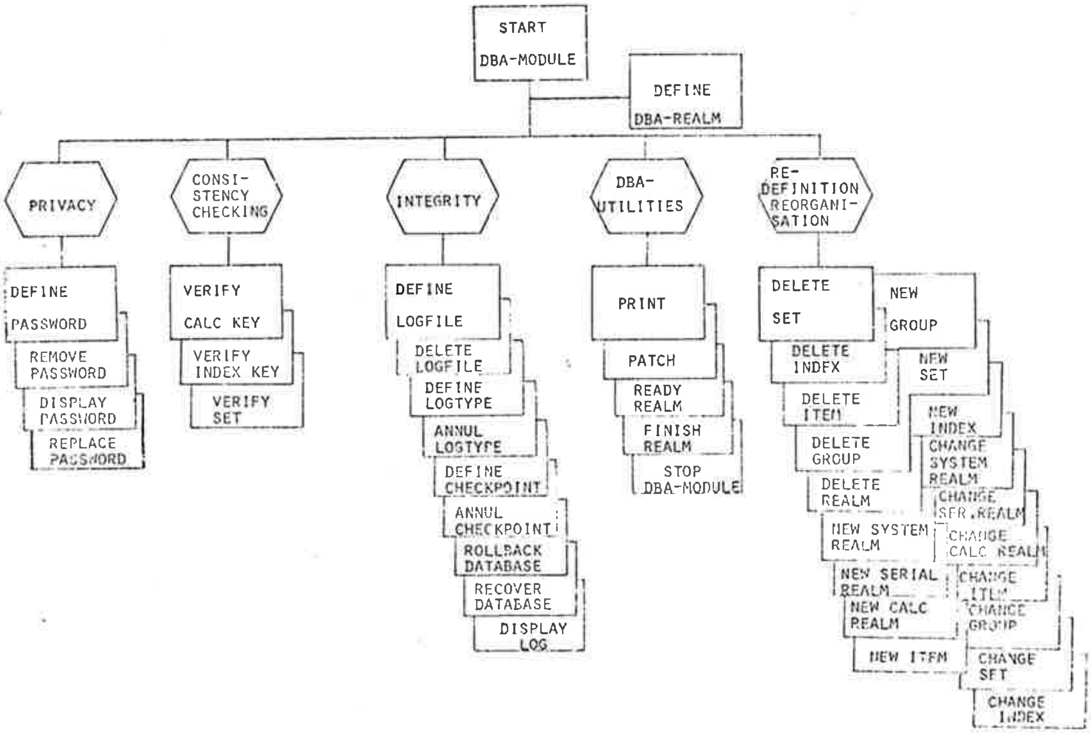
TABLE OF CONTENTS

APPENDICES

1. SIBAS DATA BASE ADMINISTRATION MODULE

1.1 INTRODUCTION

The DBA-module is a tool which enables the Data Base Administration to control the efficient and reliable use of the data base. The functions included in the DBA-module are shown in the figure below:



DATA BASE ADMINISTRATION FUNCTIONS

The use of any of the DBA-functions is controlled by the START DBA-MODULE statement. In this statement a DBA-password may be provided, and the validity of this password is checked on the actual data base. This prevents the unauthorized use of the DBA-module on a particular data base.

1.2 SYNTAX DESCRIPTION

The data-base administrator statements may be written in a format free syntax.

Throughout this manual, wherever a Data-Base Administrator statement is described, some conventions are used:

$\underline{A}$	A must be present
A	A is optional
$\left\{\begin{matrix}\underline{A}\\ \underline{B}\end{matrix}\right\}$	A or B must be present

⟨realm-name⟩ "realm-name" is a parameter which may take different values

[⟨ub⟩] the parameter "ub" is optional

Parameter values may be SIBAS names, integers, pointer values ..... See the relevant implementor's note for the convention used to code these parameters.

The messages issued by the DBA-module are self-explanatory.

2 DBA-UTILITIES STATEMENTS

2.1 START DBA-MODULE

Function

The function of this statement is to indicate the user's intention to process data-base administrator statements and to check that the user is allowed to do so.

Syntax

```
START  DBA-MODULE FOR  DATABASE <data-base-name>
      [DBA-PASSWORD <dba-password>].
```

Rules

1. "data-base-name" is the name of the data-base as is given in OPEN DATA-BASE.

2. If privacy is defined for the data-base, the "dba-password" will be checked to decide whether or not the user is allowed to process dba statements.

3. The effect of this statement is to physically open the data-base.

2.2 STOP DBA-MODULE

Function

To prevent the further processing of data-base administrator statements apart from START DBA-MODULE.

Syntax

STOP DBA-MODULE.

Rules

1. The effect of this statement is to physically close the data-base.

2. Realms previously readied with READY statement are automatically finished by STOP.

2.3 DEFINE DBA-REALM

Function

Define the data-base administrator realm which is the realm upon which the passwords and log descriptions are stored.

Syntax

DEFINE DBA-REALM <realm-name> SIZE<size>.

Rules

1. "realm-name" must be different from any existing realm name in the data-base schema.

2. "size" is the maximum number of passwords to be defined for the database.

3. The realm must have been previously assigned by the operating system. It's size in words should be approximately the number of passwords x 510.

2.4 READY REALMS

Function

This statement indicates to the DBA-MODULE the user's intention to process records on one or more realms.

Syntax

$$\underline{\text{READY}} \begin{Bmatrix} \underline{\text{ALL}} \\ \underline{\text{REALM}} \quad \langle\text{realm-name}\rangle \end{Bmatrix} .$$

Rules

1. The effect of this statement is to ready the realm "realm-name" or all the realms in the data-base for exclusive-update.

2. This statement must be successfully executed before any PRINT, PATCH or VERIFY statement may be executed.

2.5 FINISH REALMS

Function

To prevent further processing of the data on one or all realms.

Syntax

$$\underline{\text{FINISH}} \left\{ \begin{array}{ll} \underline{\text{REALM}} & \langle\text{realm-name}\rangle \\ & \underline{\text{ALL}} \end{array} \right\}.$$

Rules

1. The effect of this statement is to prevent further use of the referred realms for PRINT, PATCH or VERIFY.

2. The STOP statement automatically finishes all realms.

2.6 PRINT

Function

To print the content of the specified units of information in a formatted dump form.

Syntax

```
                              ⎧ BUCKET ⎫          ⎧        POINTER  <address>    ⎫
PRINT  │    ALL     │         ⎪ PAGE   ⎪  FROM    ⎨ [<unit-nr>] REALM            ⎬
       │ <number>   │         ⎨ RECORD ⎬          ⎪        <realm-name>          ⎪
                              ⎩ WORD   ⎭          ⎩                              ⎭

       PRINT       POINTER     <address> .
```

Rules

1. "number" is an integer. If neither "number" nor ALL are specified, it is assumed that "number" is equal to 1.

2. FROM may specify a data-base address or a word-address within the realm "realm-name".

3. When RECORD is specified all records within the defined range are printed, deleted records as well as active records.

4. All the realms involved must be readied prior to PRINT.

5. "unit-nr" specifies the start for the dump as a BUCKET PAGE, RECORD, or WORD number.

6. "address" may be specified in decimal or octal (NORD, UNIVAC) or hexadecimal (IBM). An "address" starting with 0 (zero) will be treated as octal or hexadecimal. Pointer address specification is machine dependant, see implementors notes.

2.7 PATCH

Function

To Replace one word in the data-base.

Syntax

PATCH <word-nr> REALM <realm-name>

REPLACE <old-value> WITH <new-value>.

Rules

1. The use of this statement implies a very good knowledge of how a SIBAS data-base is built up internally and should only be used in extreme cases.

2. "word-nr" is the address of the word to be patched in the realm "realm-name".

3. "old-value" and "new-value" may be specified as decimal numbers, octal numbers or hexadecimal numbers (IBM). Numbers starting with 0 (zero) are treated as octal/hexadecimal.

4. Since no logging takes place while the DBA-module is under execution, it may be necessary to take new copies of all or part of the data base after use of the PATCH function.

3. PRIVACY

3.1 GENERAL

The privacy system enables the DBA to restrict the use of the data base to authorized users. This is done by defining passwords for the data base or a part of the data base, and connecting the actual usage mode to each password. Privacy can be defined on three levels:

1. Privacy on the data base level.
2. Privacy on the realm level.
3. Privacy on the record occurrence level.

The privacy functions of the DBA-module are used to define and give values to passwords on the data base and the realm level (fig. 2.1). Either the Data Definition Language or the Redefinition Language is used to define privacy on the record occurrence level, and the Data Manipulation Language is used to give values to the privacy items in each record occurrence (fig. 2.2).

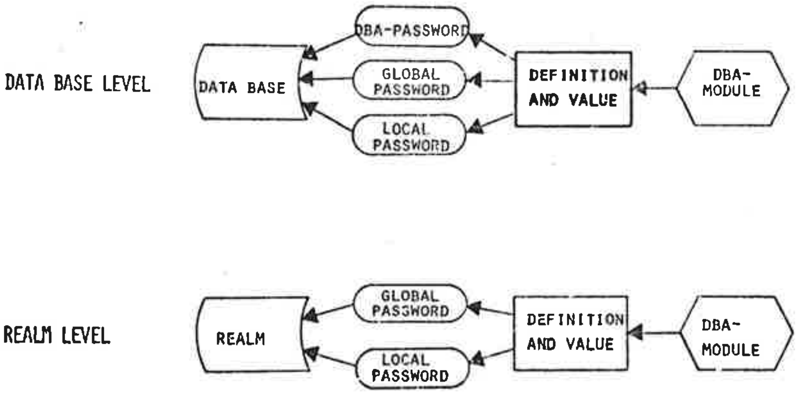


Fig. 2.1: Defining and giving values to passwords on the Data Base and the Realm level.

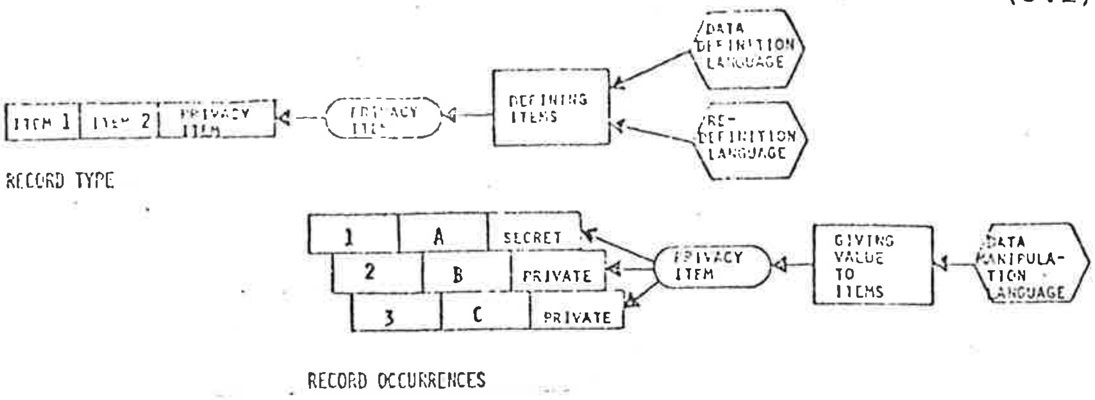


Fig. 2.2: Defining and giving value to Privacy Items on the Record Occurrence level.

If privacy is defined on the Data Base Level for a data base, each run unit must give in a password with the OPEN DATA BASE statement. This validity of password will be checked and if it is valid it will remain "current password" for the run unit until CHANGE PASSWORD is used to update the current password.

If privacy is defined on the Realm Level, validity of each run unit's current password will be checked when READY REALM is executed. The current password for the run-unit must be valid for the USAGE MODE and PROTECTION MODE given in the READY REALM statement.

Privacy cannot be defined on system Realms. If privacy is defined on the Record Occurrence Level, each run unit's current password will be checked for validity when the run unit attempts to execute a MODIFY, ERASE, CONNECT, INSERT or GET on a record. The current password must match the value of the Privacy Item in the record. In case of ERASE, all records to be erased in a single ERASE statement are checked.

Before privacy is defined at the Realm or Data Base Level, a DBA-password may be defined in addition to other passwords. The DBA-password will allow a user (Data Base Administrator) to execute START DBA-MODULE, and to perform any of the functions included in the DBA-MODULE and any DML statement.

Table 2.1 shows how privacy restrictions on a data base are defined, how and when passwords may be defined and modified, and when the privacy checks are performed by the SIBAS run-time control system (DBCS).

TYPE OF PRIVACY	HOW PRIVACY IS DEFINED	HOW PASSWORDS ARE GIVEN VALUES	HOW PASSWORDS ARE MODIFIED	WHEN THE VALIDITY OF CURRENT PASS-WORD IS CHECKED
DBA-PASSWORD	USING DBA MODULE	USING DBA MODULE	USING DBA MODULE	AT EXECUTION OF OPEN DATABASE READY REALM START DBA-MODULE
DATA-BASE LEVEL	USING DBA MODULE	USING DBA MODULE	USING DBA MODULE	AT EXECUTION OF OPEN DATABASE READY REALM
REALM LEVEL	USING DBA MODULE	USING DBA MODULE	USING DBA MODULE	AT EXECUTION OF READY REALM
RECORD OCCURENCE LEVEL	USING 1) SCHEMA DATA DEFINITION LANGUAGE REDEFINITION LANGUAGE	WHEN A RECORD OCCUR-RENCE IS STORED	WHEN A RECORD OCCUR-RENCE IS MODIFIED	AT EXECUTION OF MODIFY GET ERASE CONNECT/DISCONNECT INSERT/REMOVE

Table 2.1 Defining and controlling Passwords.

3.2 LOCAL AND GLOBAL PASSWORDS

Passwords defined on the Data Base Level or on the Realm Level can be either local or global.

A local password on the Data Base Level is valid for OPEN DATA BASE only. If privacy is defined on Realm and/or Record Occurrence level a new current password may have to be given before READY, GET, MODIFY, ERASE, CONNECT, DISCONNECT, INSERT or REMOVE can be executed.

A global password on the Data Base Level is valid for OPEN DATA BASE. In addition it will allow the run unit to execute READY REALM, with the USAGE MODE and PROTECTION MODE defined for the password, on any realm in the data base. It will also allow the run unit to execute other dml-statements, regardless of the value of the privacy item in each record (MODIFY and ERASE can only be executed if the realm was readied with USAGE MODE UPDATE: CONNECT, DISCONNECT, INSERT, REMOVE, can be executed if the realm was readied with USAGE MODE LOAD.)

A local password on Realm level is valid only for READY REALM with the USAGE MODE and PROTECTION MODE defined for the password.

A global password on Realm level will in addition allow the run unit to execute dml-statements, regardless of the value of the privacy item in each record in the specified realm.

The DBA-PASSWORD is a global password on the database level with usage mode UPDATE and protection mode EXCLUSIVE.

3.3 USAGE MODE AND PROTECTION MODE

USAGE MODE and PROTECTION MODE must be defined for all passwords which allow a user to execute READY REALM, i.e. for global passwords on the Data Base level and global and local passwords on the Realm level. The possible USAGE MODES are: RETRIEVAL, LOAD and UPDATE. The possible PROTECTION MODES are: NON-PROTECTED and EXCLUSIVE.

Table 2.2 gives a summary of the functions allowed for different types of password, assuming that privacy is defined on all three levels.

TYPE OF PASSWORD			DBA - PASSWORD	DATA BASE LEVEL							REALM LEVEL													RECORD OCCURRENCE LEVEL
				LOCAL	GLOBAL						GLOBAL						LOCAL							
					NON-PROTECTED			EXCLUSIVE			NON-PROTECTED			EXCLUSIVE			NON-PROTECTED			EXCLUSIVE				
FUNCTION					RETRIEVAL	LOAD	UPDATE	RETRIEVAL	LOAD	UPDATE	RETRIEVAL	LOAD	UPDATE	RETRIEVAL	LOAD	UPDATE	RETRIEVAL	LOAD	UPDATE	RETRIEVAL	LOAD	UPDATE		
OPEN DATA BASE			X	X	X	X	X	X	X	X													
READY REALM	NON-PROTECTED	RETRIEVAL	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
		LOAD	X			X	X		X	X		X	X		X	X		X	X		X	X	
		UPDATE	X				X			X			X			X			X			X	
	EXCLUSIVE	RETRIEVAL	X					X	X	X				X	X	X				X	X	X	
		LOAD	X						X	X					X	X					X	X	
		UPDATE	X							X						X						X	
STORE			X			X	X		X	X		X	X		X	X		X	X		X	X	
GET			X		X	X	X	X	X	X	X	X	X	X	X	X							X
MODIFY, ERASE ELEMENT			X				X			X			X			X							X
ERASE			X				X			X			X			X							X
INSERT, REMOVE CONNECT, DISCONNECT			X			X	X		X	X		X	X		X	X							X
START DBA-MODULE			X																				

Table 2.2: Functions allowed for different types of privacy. It is assumed that privacy is defined at all levels.

3.4 SUMMARY OF THE SETTING OF CURRENT PASSWORD

Initially the current password is set for a run-unit when the data base is opened. Unless a CHANGE PASSWORD statement is performed, the value of the current password will remain unchanged. When a READY REALM statement is performed, the current password must match a password which is defined for the desired mode of operation on the realm. If the run-unit performs a data manipulation statement on records where the value of the privacy item is different from the realm password, the current password for the run-unit must be changed to match the value of the privacy item before the data manipulation statement is successfully executed.

3.5 DEFINE PASSWORD

Function

The function of this statement is to register a new password. Passwords can be of five different types, and for three of them USAGE MODE and PROTECTION MODE is given with the password.

Syntax

This statement has 5 different formats, one for each password type.

```
1.  DEFINE      DBA-PASSWORD      <dba-password> .

2.  DEFINE      LOCAL-PASSWORD    <password-1>    ON DATABASE  .

3.  DEFINE      GLOBAL-PASSWORD   <password-2>    ON DATABASE

       [       ⎧ RETRIEVAL ⎫ ]   [              ⎧ NON-PROTECTED ⎫ ]
       [ USAGE ⎨ LOAD      ⎬ ]   [ PROTECTION   ⎨               ⎬ ]  .
       [       ⎩ UPDATE    ⎭ ]   [              ⎩ EXCLUSIVE     ⎭ ]

4.  DEFINE      LOCAL-PASSWORD    <password-3>
       ON REALM          <realm-name-1>

       [       ⎧ RETRIEVAL ⎫ ]   [              ⎧ NON-PROTECTED ⎫ ]
       [ USAGE ⎨ LOAD      ⎬ ]   [ PROTECTION   ⎨               ⎬ ]  .
       [       ⎩ UPDATE    ⎭ ]   [              ⎩ EXCLUSIVE     ⎭ ]

5.  DEFINE      GLOBAL-PASSWORD   <password-4>
       ON REALM          <realm-name-2>

       [       ⎧ RETRIEVAL ⎫ ]   [              ⎧ NON-PROTECTED ⎫ ]
       [ USAGE ⎨ LOAD      ⎬ ]   [ PROTECTION   ⎨               ⎬ ]  .
       [       ⎩ UPDATE    ⎭ ]   [              ⎩ EXCLUSIVE     ⎭ ]
```

Rules

1. LENGTH OF PASSWORDS. All passwords must follow the same conventions as SIBAS names, i.e. up to 8 bytes, starting with a letter, no embedded blanks, but trailing blanks allowed.

2. DBA-PASSWORD. When a "dba-password" is defined for a data base it must always be given with the START DBA-MODULE statement. The "dba-password" will also serve as a GLOBAL-PASSWORD on a DATABASE with USAGE UPDATE and PROTECTION EXCLUSIVE. This implies that the "dba-password" also allows one to execute any DML-statement in addition to the START DBA-MODULE.

3. LOCAL-PASSWORD ON DATABASE. "password-1" will serve as a local password on the data base level. The validity of this password is restricted to the OPEN DATABASE statement.

4. GLOBAL PASSWORD ON DATABASE. "password-2" will serve as a global password on the data base level. In addition its use with the OPEN DATABASE statement, the password will be valid for the execution of READY REALM with the USAGE mode and PROTECTION mode given, and for the execution of any other DML-statements covered by the usage mode.

5. LOCAL PASSWORD ON REALM. "password-3" will serve as a local password for the realm given in "realm-name-1". It will be valid for executing READY REALM on the realm given in "realm-name-1" with the USAGE mode and the PROTECTION mode given. "password-3" will not give admission to execute other dml-statements on records in the realm if privacy on record occurrence level is defined.

6. GLOBAL PASSWORD ON REALM. "password-4" will serve as a global password for the realm given in "realm-name-2". It will be valid for executing READY REALM on the realm given in "realm-name-2" with the USAGE mode and PROTECTION mode given. In addition it will give admission to execute any other DML-statement on the records in the realm covered by the given USAGE mode.

7. IDENTICAL PASSWORDS. Two passwords defined on data base level may be identical if one is local and the other is global. Two passwords defined on realm level for a particular realm may be identical if one is local and the other is global. Passwords defined for different realms may be identical (e.g. a password on data base level may be identical with a password for one or more realms and identical with the privacy item in one or more record occurrences). Using the same password on different levels means that a run-unit may not have to change current password, and this gives the effect of a "global" password for a part of the data base.

8. USAGE and PROTECTION are optional. If USAGE is not given, RETRIEVAL is assumed. If PROTECTION is not given, NON-PROTECTED is assumed.

9. MULTIPLE PASSWORDS. two or more passwords with different values may be defined with the same specification, i.e. same type, same realm, same protection and usage.

3.6 REMOVE PASSWORD/PRIVACY

Function

The function of this statement is to remove a single password defined on realm or data base level or to remove all privacy defined on the realm or data base level.

Syntax

$$\underline{\text{REMOVE}} \left\{ \begin{array}{l} \underline{\text{PASSWORD}} \ \langle \text{password} \rangle \\ \underline{\text{PRIVACY}} \end{array} \right\} \left[\text{FROM} \quad \left\{ \begin{array}{l} \underline{\text{DATABASE}} \\ \underline{\text{REALM}} \langle \text{realm-name} \rangle \end{array} \right\} \right].$$

Rules

1. REMOVE PASSWORD " password" FROM DATABASE. This operation will remove the password given in "password" from the list of passwords on data base level.

2. REMOVE PASSWORD "password" FROM REALM "realm-name". This option will remove the password given in "password" from the list of passwords defined for the realm given in "realm-name".

3. REMOVE PASSWORD "password". Every occurence of "password" is removed from database description.

4. REMOVE PRIVACY FROM DATABASE. This option will remove all passwords defined on data base and realm level. It will also remove the DBA-password.

5. REMOVE PRIVACY FROM REALM "realm-name". This option will remove all passwords defined for the realm given in "realm-name".

6. REMOVE PRIVACY. All passwords defined are removed from database description, including the DBA-password.

3.7 DISPLAY PASSWORD/PRIVACY

Function

The function of this statement is to print the values and the description of all or some valid passwords.

Syntax

```
        { ALL PRIVACY            }
DISPLAY {                        } .
        { PASSWORD <password>    }
```

Rules

1. If the ALL option is given a complete report is printed containing the values of all passwords defined for the data base. The report will also contain the type, usage mode and protection mode for each password.

2. If the PASSWORD option is given the type(s), usage mode and protection mode for the password specified are given. All definitions of the password with the given value will be printed.

3.8 REPLACE PASSWORD

Function

The function of this statement is to replace the value of a password with a new value for all occurrences of the password (i.e. one at data base level and one or more at realm level).

Syntax

REPLACE ⟨password-1⟩ WITH ⟨password-2⟩ .

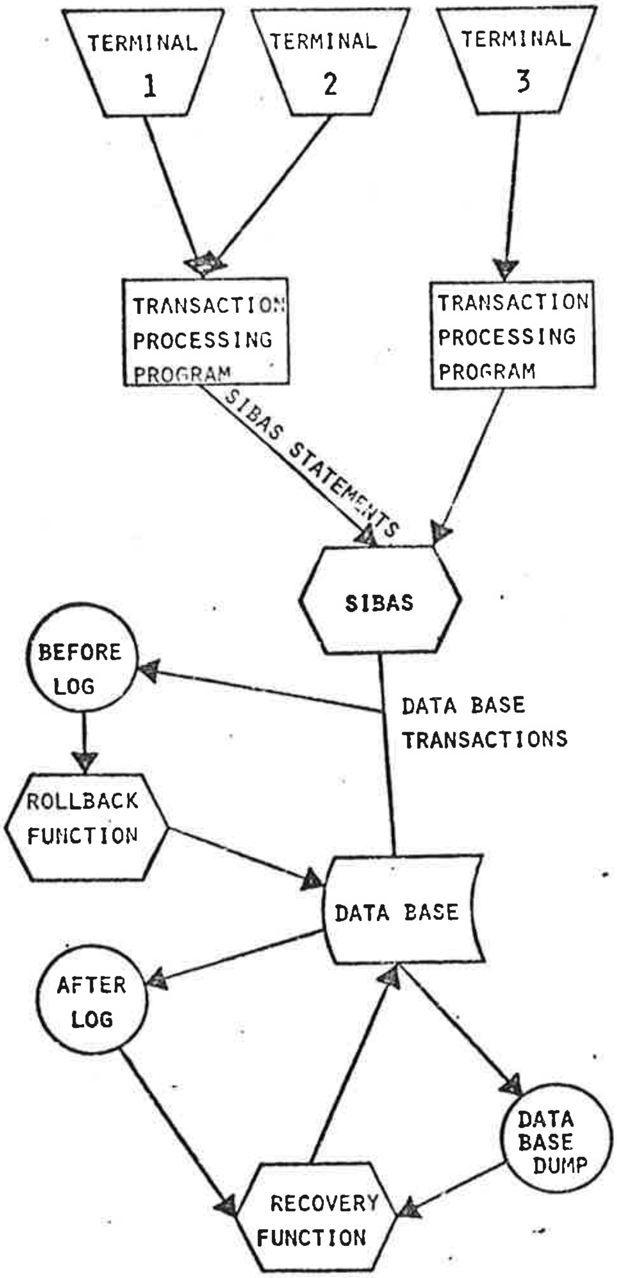
Rules

1. The value given in "password2" must not be equal to any already defined password.

4. SIBAS INTEGRITY SYSTEM

4.1 INTRODUCTION

The SIBAS integrity system includes functions for logging, rollback, recovery, checkpointing and the initiation and termination of transactions.



Checkpointing

In SIBAS a method of checkpointing is used, which quiesces the system at frequent intervals so that no transactions are active. If a failure occurs, it is possible to reconstruct the data base to a state consistent with the transactions completed at the latest checkpoint. When a checkpoint is taken, the contents of the SIBAS buffer area will be written to the data base.

Logging

Logging involves copying to a log file all pages which are written from the SIBAS buffer area to the data base. The logging can be done before the pages are updated, resulting in an "old copy audit trail", or after the pages are updated resulting in a "new copy audit trail". The log files can be tape files or direct access files. It is also possible to log the Data Manipulation calls.

Rollback

In case of a program fault, the execution of a transaction may stop without closing the realms properly, i.e. possible updates on the data base may not have been written from the buffer area to the data base. In this case the rollback function can be used together with the "before look" (old copy audit trail) to bring the data base back to the state of the last or previous checkpoint.

Recovery

When serious faults occur on the data base, it may not be possible to recover using the rollback function (e.g. in the case of a disc fault). In this case the recovery function must be used. The recovery function uses a complete dump of the data base and the "after look" (new copy audit trail). The database dump is updated with the logged pages up to a specified checkpoint and results in a consistent version of the data base.

Dump

Logging is turned off when one is using the DBA-module. If the database is changed during this DBA-module use, a full copy ("Dump") of the database should be made.

4.2 DEFINE LOGFILE

Function

The function of this statement is to define a new file on which SIBAS logs may be written.

Syntax

```
                                                    ⎧ TAPE ⎫
    DEFINE LOG-FILE  <filename>  MEDIUM ⎨ DISC ⎬
                                                    ⎩ DRUM ⎭

    FILE-SIZE      <length-1> RESERVED-LENGTH    <length-2>
 ⎡⎧ BLOCK-GAP   ⎫                 ⎤
 ⎢⎨             ⎬ <length-3>      ⎥ .
 ⎣⎩ SECTOR-SIZE ⎭                 ⎦
```

Rules

1. FILENAME. The "file-name" must be the name of a file defined to the operating system. No other logfile with the same name must exist.

2. FILESIZE. The "length-1" must contain the length of the logfile in computer words.

3. RESERVED-LENGTH. The "length-2" must contain the number of words reserved on the logfile. When there are "length-2" words left in the logfile, the logfile is treated as full, and a checkpoint is requested. Logging will, however, continue after the checkpoint is taken until the logfile is completely filled. It is the DBA's responsibility that enough space is defined between "length-1" and "length-2" to hold all the logs created by the checkpoint. If there is no other log file to receive new page logs, the execution is terminated.

4. BLOCK-GAP/SECTOR-SIZE. The BLOCK-GAP/SECTOR-SIZE option must be given for logfiles on respectively tape and disk. The "length-3" must be the block gap or sector size in computer words. The default value for sector size is 128 words. The block-gap value is used to compute how much of the log-file is actually useful.

5. MAXIMUM NUMBER OF LOGFILES. The maximum number of logfiles defined at the same time is two. Different logtypes may be mixed on the same log-file.

6. Definition of a log-file on DISC/DRUM has the effect of physically zeroing the file. If a used log-file is deleted and defined again, the content of the file is lost.

4.3 DELETE LOGFILE

Function

The function of this statement is to remove the definition of an existing logfile.

Syntax

DELETE   LOG-FILE <filename>

Rules

1. FILENAME. "filename" must refer to a file defined as a logfile for the database.

2. DEFINED LOGTYPES. All log-types defined for the logfile given must be annulled prior to this statement.

3. EFFECT OF DELETE. The execution of this statement does not erase the content of the logfile. The effect is only to make the file unknown to the DBA-module.

4.4 DEFINE LOG-TYPE

Function

The function of this statement is to define the types of log which will be taken automatically by the system, and what types of information the user will be allowed to log.

Syntax

```
                                                BEFORE-LOOK
                                               ⎧            ⎫
                            ⎧ DATABASE         ⎨ AFTER-LOOK ⎬      ⎫
                            ⎪                  ⎩ BOTH       ⎭      ⎪
                            ⎪                                      ⎪
DEFINE      LOG-TYPE        ⎨ DML-STATEMENT                        ⎬ LOG-FILE
                            ⎪                                      ⎪ <file-name> .
                            ⎪                      TRANSACTION     ⎪
                            ⎪                    ⎧                 ⎫
                            ⎩ USER-INITIATED     ⎨ DML-STATEMENT   ⎬
                                                 ⎩ BOTH            ⎭
```

Rules

1. FILENAME. The "filename" must previously have been defined by use of DEFINE LOGFILE. A logfile may be used to hold different types of logs.

2. AFTER-LOOK. If the AFTER-LOOK option is given, each page which has been updated will be copied to the log after it has been written back to the data base. It is important to note that many users may have updated the same page in the buffer before it is written back to the data base and the log is taken. This log-type is used by the RECOVERY function (section 4.9).

3. BEFORE-LOOK. If the BEFORE-LOOK option is given, each page which is to be updated is copied from the database to the logfile before the updated version is written to the data base. The log will then contain a picture of the database before each update. The BEFORE-LOOK log is used by the ROLL-BACK function (section 4.8).

4. BOTH. If the BOTH option is given, both AFTER-LOOK and BEFORE-LOOK logs will be taken on the logfile given in "file-name".

5. DML-STATEMENT. If this option is given all DML-statements executed except ACCEPT will be copied to the logfile given in "file-name". With each DML-statement the value of each input parameter is logged. This log-type is the basis for a recovery procedure, in addition to the already existing RECOVERY function (section 4.9). This log-type is also useful for tracing and debugging, and it might serve as a basis for an automatic restart procedure.

6. USER-INITIATED TRANSACTION. If this option is given, all users will be allowed to use the LOG-statement in the DML to write any information to the log during execution of a SIBAS PROGRAM. The LOG-statement is described in ch. 5 of the USERS MANUAL.

7. USER-INITIATED DML-STATEMENT. If this option is given, each user will be allowed to use the DML LOG-statement to log individual DML-statements during execution of a SIBAS program. The use of this statement is described in ch. 5 of the USERS MANUAL.

8. USER-INITIATED BOTH. If this option is given, all users will be allowed to use both the formats of the LOG-statement (see ch. 5 in the USERS MANUAL).

4.5 ANNUL LOG-TYPE

Function

The function of this statement is to remove a log-type from a given logfile.

Syntax

```
                          ⎧           ⎧ BEFORE-LOOK ⎫             ⎫
                          ⎪ DATABASE  ⎨ AFTER-LOOK  ⎬             ⎪
                          ⎪           ⎩ BOTH        ⎭             ⎪
ANNUL  LOG-TYPE           ⎨ DML-STATEMENT                         ⎬ LOG-FILE
                          ⎪                                       ⎪  <file-name>
                          ⎪                  ⎧ TRANSACTION   ⎫    ⎪
                          ⎪ USER-INITIATED   ⎨ DML-STATEMENT ⎬    ⎪
                          ⎩                  ⎩ BOTH          ⎭    ⎭
```

Rules

1. LOGFILE. The file identified by "file-name" must have been defined as logfile for one or more of the log-types.

2. EXISTING LOGS. The logs which are already written to the logfile will not be deleted when a log-type is annulled.

3. ANNUL of a log-type which was not previously defined is ignored.

4.6 DEFINE CHECKPOINT

Function

The function of this statement is to define the types of checkpoint to be taken automatically by the system, and to give permission to the users to request checkpoints.

Syntax

```
DEFINE  CHECKPOINT  [ MAXLOG <size>    ]  [ SIGN-OFF ] [ USER ]
          LOG-FILE  [ <file-name> .    ]
```

Rules

1. FILENAME. The "filename" must previously have been defined by use of DEFINE LOGFILE. More than one checkpoint-type may be defined on the same logfile.

2. MAXLOG. For each time "size" words are written on the logfile a checkpoint is requested.

3. SIGNOFF. Whenever a user executes a close database statement a checkpoint is requested.

4. USER. If this option is specified all users are allowed to request checkpoints from their user programs. This is done by use of the CHECKPOINT statement described in ch. 5 of the USERS MANUAL.

5. MULTIPLE LOG-FILES. Every time a checkpoint is requested, a checkpoint is written on every defined log-file, i.e. the checkpoint system is synchronized.

6. FUNCTION OF CHECKPOINTING. A request for checkpoint is not executed until all users have terminated their active transactions. New transactions are rejected until the checkpoint is taken. When all user transactions are terminated the internal record buffers are written to the relevant realms and a checkpoint record is written on each logfile.

7. AUTOMATIC CHECKPOINTS. Checkpoint are automatically requested on all defined logfiles when the database is physically opened or closed. When one logfile is filled up to the reserved area, a checkpoint is requested.

4.7 ANNUL CHECKPOINT

Function

The function of this statement is to annul the taking of automatic or user initiated checkpoints.

Syntax

```
ANNUL  CHECKPOINT  [MAXLOG] [SIGN-OFF] [USER]  LOG-FILE
                                                <file-name>
```

Rules

1. FILENAME. the "file-name" must be the name of a file defined as a logfile.

2. MAXLOG. If this option is specified checkpoint on maxlog words will not be taken.

3. SIGNOFF. If this option is specified checkpoint on signoff will not be taken.

4. USER. In this case the users are no longer allowed to request checkpoints.

5. If the option to be annulled is not previously defined, no error message is given.

4.8 ROLLBACK

Function

The function of this statement is to re-establish the database state to a given or to the previous checkpoint from the existing database using before-look logs.

Syntax

```
                                  ┌ LAST                                ┐
ROLL-BACK   DATABASE   TO        {                    <checkpoint-id>   }
                                  └ CHECKPOINT                          ┘

        LOG-FILE <file-name-1>        [ ADJUST   OTHER   LOG-FILE ] .
```

Rules

1. FILENAME. The "file-name-1" must previously have been defined by use of DEFINE LOGFILE. Log of type BEFORE-LOOK must have been defined.

2. FUNCTION. The database is rolled back, i.e. the logfile "file-name-1" is read backwards and the database is updated each time a before-look log page is found. The process is stopped either when a checkpoint is found if LAST is specified, or when a checkpoint is found with a time and date referring to the same or an earlier identification than "checkpoint-id". If no match is found when the start of the logfile is reached, an error message is given. The database will, however, be consistent at this point because all logfiles are started and ended with a checkpoint. If the logfile is distributed on more than one physical file, the process may be continued from this point by giving a new ROLLBACK statement.

3. I/O-ERROR. If an I/O-ERROR occurs during the rollback, the process is stopped and a SIBAS core dump results.

4. ADJUST. If two logfiles are specified, the unused logfile should be adjusted to the same checkpoint as the actually used logfile. This may be done with the ADJUST option.

5. CHECKPOINT IDENTIFICATION. Consists of the time and date followed by a sequence number.

4.9 RECOVER

Function

The function of this statement is to re-establish the database state to a given checkpoint using a database dump taken at a specified checkpoint and after-look logs.

Syntax

```
RECOVER   DATABASE
TO <checkpoint-id>            LOG-FILE     <file-name-1>
        [ADJUST   OTHER       LOG-FILE ] .
```

Rules

1. FILENAME. The "file-name-1" must have been previously defined by use of DEFINE LOGFILE. For "file-name-1" log of type AFTER-LOOK must have been defined.

2. CHECKPOINT. The database is recovered, i.e. the logfile "file-name-1" is read forward until a checkpoint consistent with the dump is found. The reading then continues and for each after-look log page found the database is updated. The process continues until a checkpoint equal "checkpoint-id" is found. If no checkpoint consistent with the dump is found, an error message is given. In this case the database will not be updated.

   If "checkpoint-id" is not found before the end-of-file, an error message is given. In this case the data base will be updated and it will also be consistent. The recovery process may continue with a new logfile (or a new tape) by giving the RECOVERY statement once more.

3. I/O-ERROR. If an I/O-ERROR occurs during recovery the process is stopped and a SIBAS core dump results.

4. ADJUST. If two logfiles are specified the unused logfile should be adjusted to the same checkpoint as the actually used logfile. This may be done with the ADJUST option.

5. CHECKPOINT IDENTIFICATION. Consists of the time and date followed by a sequence number.

4.10 DISPLAY LOG

Function

Print the description of the log files and/or the types of log defined on the database.

Syntax

```
        ( LOG       )
DISPLAY <           >  .
        ( LOG-TYPE  )
```

Rules

1. If the LOG-TYPE option is given - a report is printed showing the type of log defined for the database and the identification of the last checkpoint taken.

2. If the LOG option is given, the description of the log-file is printed together with it's current state. Information about log-types is also printed.

5. CONSISTENCY CHECKING

5.1 GENERAL

The consistency checking function are a part of the integrity control system for the database. These functions are used to detect integrity breaches. When breaches on the database integrity are detected, the recovery system will normally be used to bring the database back to a consistent state. In some cases the patch functions can be used to do minor repairs on the database.

It should be noted that consistency checking does not include validity checking. Validity checking is concerned with the logical content of the database as viewed by the user, consistency checking is concerned with the physical content of the database and its consistency vis a vis the database's physical construction.

The types of consistency checking which can be performed in SIBAS are:

- CALC KEY verification
- INDEX KEY verification
- SET verification

No attempt is made by the consistency processor to correct breaches.

If breaches are detected, the following information will be given:

Message:
"Message describing the type of breach"

Information about the record:
"Realm name", "Item name"
"Physical position of the record (pointer)"
"Item value"
"Comparing value"
"Dump of record"

It must be noted that the item name can be the name of a pointer (see Record layout printed from DDL processor).

If any syntax error is detected, a message, describing the type of error, is printed.

All realms to be verified must be readied..

A verify run may look like:

```
START     DBA-MODULE    FOR    DATABASE    FUNCBASE.
READY     ALL.


          VERIFY   CALC    DATABASE.
          VERIFY   INDEX   DATABASE.
          VERIFY   SET     DATABASE.

FINISH    ALL.
STOP.
```

5.2 CALC KEY VERIFICATION

Function

This function provides for the verification of calc key consistency. For each calc key verification, the calc key of all records stored in the specified realm will be checked. The value of the calc key is checked against the bucket number of the record.

No attempt is made to correct errors which are detected by calc key verification. Information about the record and its physical position is printed.

Syntax

$$\underline{\text{VERIFY}}\ \underline{\text{CALC}} \left\{ \begin{array}{ll} \underline{\text{DATABASE}} & \\ \underline{\text{REALM}} & \langle\text{realm-name}\rangle \end{array} \right\}$$

$$\left[\underline{\text{MAXREC}}\ \text{OF}\langle\text{integer}\rangle \right] .$$

Rules

1. DATABASE. If the DATABASE option is given, all calc keys on the database will be checked.

2. REALM.If the REALM option is given, the calc key on the specified realm will be checked.

3. MAXREC.If the MAXREC option is used, the verification process will stop when "integer" records have been checked.

4. ERROR MESSAGE. If one or more inconsistent calc keys are detected, the following information is given for each error:

CALCULATED KEY DOES NOT CORRESPOND TO RECORD KEY

"information about the record" (see 5.1)

5.3 INDEX KEY VERIFICATION

Function

This function checks the consistency of index key values and index table entries.

The command specifications allow for the checking of all index tables in the database, or specified index tables in a realm.

The function of the index key verification is to check the consistency of the key value of each entry in the index table with the corresponding key value in the record for each index key defined. The consistency checks are performed in two ways:

1) By reading all the entries in the index table and finding the corresponding record.

2) By scanning all the records and using the key value to find the corresponding table entry. This check is performed for automatically maintained indexes only.

No attempt is made to correct a detected error.

Syntax

```
             { DATABASE
VERIFY INDEX {
             { REALM   <realm-name> KEY <key-name-1>[<key-name-2

  [MAXREC OF <integer>].
```

Rules

1. DATABASE. If the DATABASE option is given, all index keys defined for the database will be checked.

2. REALM. If the REALM option is given, all index keys given in key-name-1, key-name-2 .... will be checked.

3. MAXREC. If the MAXREC option is used, the verification process will stop when "integer" records have been checked.

4. ERROR MESSAGE. The errors that may be detected are:

   1) ENTRY IN INDEX TABLE DOES NOT MATCH RECORD KEY
   2) RECORD HAS NO CORRESPONDING ENTRY IN INDEX TABLE

   Information about the record will be printed for each error detected (see 5.1).

5.4 SET VERIFICATION

Function

This function is used for verifying set relationships within the database. This function is performed by traversing records of a set and examining their types and their pointers.

The set verification utility may be requested to vary its domain of examination from a single set occurrence, to all occurrences of a specified set, to all sets of a database through the specification of the appropriate format of the VERIFY command.

The consistency checks are performed in two ways:

1) By following all chains from the owner records.

2) By scanning all the member records and using the member set item value to find an owner record. This check will be performed for automatically maintained sets only.

Syntax

The set verification command has three formats:

Format 1:

```
 VERIFY SET DATABASE [MAXREC OF<integer>].
```

Format 2:

```
 VERIFY SET<setname>[MAXREC OF <integer>].
```

Format 3:

```
 VERIFY SET<set-name>USING SET-OCCUR<owner-item-value-1>,
[<owner-item-value-2>.....]
[MAXREC OF<integer>].
```

Rules

1. DATABASE. Format 1 is used when all occurrences of all sets defined for the database are to be verified. The user is warned that the amount of processing required to accomplish such a function may be considerable.

2. ALL SET OCCURRENCES IN A SET. Format 2 is used to verify all occurrences of a given set.

3. SINGLE SET OCCURRENCES. Format 3 is used to verify specified occurrences of a given set. Each set occurrence is identified uniquely by the value of the owner set item.

4. MAXREC. The MAXREC clause is used to specify the maximum number of records to be verified.

5. ERROR MESSAGES. The errors detected may be:

   1) NO OWNER RECORD FOUND WITH GIVEN OCCURRENCE
   2) POINTER POINTS OUTSIDE SET
   3) MEMBER ITEM VALUE NOT EQUAL TO OWNER ITEM VALUE
   4) BACKWARD POINTER IS ERRONEOUS
   5) OWNER POINTS TO ITSELF
   6) MEMBER HAS NO OWNER
   7) LOOP, POINTER POINTS TO A PERVIOUS MEMBER OF SET-OCCURRENCE
   8) MEMBER HAS DIFFERENT OWNER
   9) NUMBER OF RECORDS READ VIA SET DOES NOT CORRESPOND TO NUMBER OF RECORDS READ IN PHYSICAL ORDER

For each error detected information about the record involved is printed out (see 5.1).

Chains that forms a loop containing more than 512 records cannot be detected.

For error 9, additional information is printed:

```
an algebric integer = number of records read in
                      physical order
                      - number of records read via
                      set.
```

"owner-item-value-1"

each item composing owner-item-value-1 must be given a value. If it is a character item: write is as a character string delimited by quotes.

If it is an integer item: write it as an integer number.

Example:

```
VERIFY    SET    PARTOF    USING    SET-OCCUR
     ( '   MOTOR'   5    10)
     ( '   950   '  10    0) .
```

APPENDIX A

IMPLEMENTOR NOTES ON NORD-10

Command Syntax:

The syntactical units are written on one or more lines. Each command sequence is concluded by period and carriage return (".↲ ").

HELP: No command ("↲ ") gives a list of all allowed syntactical units on current command level. Parameters are listed between parenthesis, commands without.

ABBREVIATION LOOKUP: All commands (not SIBAS names) can be abbreviated, ambigouity is however not handled. The first match is used!

OCTAL NUMBERS: All octal numbers must have 0 as first digit. Otherwise the typed number is treated as decimal.

POINTERS: All pointers contain two machine words, typed as two octal numbers separated by "x".

Ex: 000400 x 012345

B SUMMARY OF DBA STATEMENTS

```
START  DBA-MODULE FOR  DATABASE  < data-base-name >
      [ DBA-PASSWORD  < dba-password > ].


DEFINE  DBA-REALM <realm-name>  SIZE<size>.


         {                ALL          }
READY    {                             } .
         {   REALM      <realm-name>   }

         {   REALM      <realm-name>   }
FINISH   {                             } .
         {                ALL          }


                      { BUCKET }          {     POINTER  < address >     }
PRINT   [    ALL    ] { PAGE   }   FROM   { [<unit-nr>]  REALM           } .
        [ <number>  ] { RECORD }          {     < realm-name >           }
                      { WORD   }
PRINT     POINTER      <address>   .


PATCH    < word-nr >    REALM    < realm-name >

         REPLACE <old-value>  WITH  <new-value>.



STOP   DBA-MODULE.
```

```
DEFINE      DBA-PASSWORD      < dba-password >.

DEFINE      LOCAL-PASSWORD    < password-1 >  ON DATABASE  .

DEFINE      GLOBAL-PASSWORD   < password-2 >  ON DATABASE

  [       { RETRIEVAL } ]   [               { NON-PROTECTED } ]
  [ USAGE { LOAD      } ]   [ PROTECTION    {               } ]  .
  [       { UPDATE    } ]   [               { EXCLUSIVE     } ]

DEFINE      LOCAL-PASSWORD    < password-3 >
   ON REALM       < realm-name-1 >

  [       { RETRIEVAL } ]   [               { NON-PROTECTED } ]
  [ USAGE { LOAD      } ]   [ PROTECTION    {               } ]  .
  [       { UPDATE    } ]   [               { EXCLUSIVE     } ]

DEFINE      GLOBAL-PASSWORD   < password-4 >
   ON REALM       < realm-name-2 >

  [       { RETRIEVAL } ]   [               { NON-PROTECTED } ]
  [ USAGE { LOAD      } ]   [ PROTECTION    {               } ]  .
  [       { UPDATE    } ]   [               { EXCLUSIVE     } ]


REMOVE { PASSWORD <password> }  FROM { DATABASE               } .
       { PRIVACY             }       { REALM <realm-name>     }


REPLACE     < password-1 >  WITH <password-2 >  .


DISPLAY { ALL PRIVACY          } .
        { PASSWORD <password>  }
```

```
                                                  TAPE
DEFINE LOG-FILE <filename>    MEDIUM            { DISC   }
                                                  DRUM

      FILE-SIZE      < length-1>   RESERVED-LENGTH    <length-2>
  [ { BLOCK-GAP   }  <length-3 > ].
    { SECTOR-SIZE }

                                   BEFORE-LOOK
                        {DATABASE {AFTER-LOOK    }       }
                        {          BOTH                  }
                        {                                }
DEFINE    LOG-TYPE      {DML-STATEMENT                   } LOG-FILE
                        {                                }   <file-name>  .
                        {                 TRANSACTION    }
                        {USER-INITIATED { DML-STATEMENT }}
                                          BOTH


DELETE    LOG-FILE <filename> .


                                  BEFORE-LOOK
                      {DATABASE { AFTER-LOOK    }       }
                      {           BOTH                  }
                      {                                 }
ANNUL   LOG-TYPE      {DML-STATEMENT                    }LOG-FILE
                      {                                 }<file-name>   .
                      {                  TRANSACTION    }
                      {USER-INITIATED  { DML-STATEMENT }}
                                         BOTH


          LOG
DISPLAY {             } .
          LOG-TYPE
```

```
DEFINE  CHECKPOINT  [MAXLOG <size>]  [SIGN-OFF] [USER]
            LOG-FILE  <file-name> .

ANNUL  CHECKPOINT  [MAXLOG] [SIGN-OFF] [USER]  LOG-FILE
                                               <file-name>  .

RECOVER   DATABASE
TO <checkpoint-id>            LOG-FILE     <file-name-1>
      [ ADJUST    OTHER       LOG-FILE ] .

                            { LAST                                 }
ROLL-BACK   DATABASE   TO   {                                      }
                            { CHECKPOINT        <checkpoint-id>    }
      LOG-FILE <file-name-1>   [ADJUST    OTHER    LOG-FILE]  .

             { DATABASE                 }
VERIFY CALC  {                          }
             { REALM    <realm-name>    }

   [MAXREC OF<integer>] .

              { DATABASE                                                              }
VERIFY INDEX  {                                                                       }
              { REALM    <realm-name>  KEY  <key-name-1>  [<key-name-2>...]           }
  [MAXREC OF<integer>].

VERIFY SET DATABASE [MAXREC OF<integer>] .

VERIFY SET<setname>[MAXREC OF<integer>].

VERIFY SET<set-name>USING SET-OCCUR<owner-item-value-1>,
[<owner-item-value-2 >....]
[MAXREC OF<integer>] .
```

NORSK DATA A.S.
Lørenveien 57 - Postboks 163, Økern
OSLO 1

COMMENT AND EVALUATION SHEET

ND-60.097.01 DATABASE ADMINISTRATOR MODULE

In order for this manual to develop to the point where it best suits your needs, we must have your comments, corrections, suggestions for additions, etc. Please write down your comments on this pre-addressed form and post it. Please be specific wherever possible.

FROM ______________________________

____

____