HP NonStop SQL/MX Comparison Guide for SQL/MP Users

Abstract

This guide describes SQL differences between HP NonStop™ SQL/MP and NonStop SQL/MX.

Product Version

NonStop SQL/MX Releases 2.0 and 2.1

Supported Release Version Updates (RVUs)

This publication supports G06.23 and all subsequent G-series RVUs until otherwise indicated by its replacement publication.

Part Number	Published
523735-003	August 2005

Document History

Part Number	Product Version	Published
429827-001	NonStop SQL/MX Release 1.0	March 2001
429857-001	NonStop SQL/MX Release 1.8	December 2002
523735-001	NonStop SQL/MX Release 2.0	June 2004
523735-002	NonStop SQL/MX Release 2.0	August 2004
523735-003	NonStop SQL/MX Release 2.1	August 2005

HP NonStop SQL/MX Comparison Guide for SQL/MP Users

Index | Figures | Tables

What's New in This Manual

Manual Information

Abstract

This guide describes SQL differences between HP NonStop™ SQL/MP and NonStop SQL/MX.

Product Version

NonStop SQL/MX Releases 2.0 and 2.1

Supported Release Version Updates (RVUs)

This publication supports G06.23 and all subsequent G-series RVUs until otherwise indicated by its replacement publication.

Part Number	Published
523735-003	August 2005

Document History

Part Number	Product Version	Published
429827-001	NonStop SQL/MX Release 1.0	March 2001
429857-001	NonStop SQL/MX Release 1.8	December 2002
523735-001	NonStop SQL/MX Release 2.0	June 2004
523735-002	NonStop SQL/MX Release 2.0	August 2004
523735-003	NonStop SQL/MX Release 2.1	August 2005

New and Changed Information

This table summarizes new and changed information for this edition of the manual:

Section	Topic	New or Changed Information
3	Expressions and Functions	Corrected information on CURRENT and EXTEND.
	LIKE Predicate	Corrected examples.
4	Similarity Checks and Automatic Recompilation	New subsection.

About This Guide

This guide describes SQL differences between NonStop SQL/MP and NonStop SQL/MX.

This release of NonStop SQL/MX enables applications to use the SQL/MX high-performance database compiler and executor to access SQL/MP and SQL/MX databases. You access tables, views, indexes, and catalogs by using SQL/MX Data Manipulation Language (DML) statements.

Audience

This guide is intended for database administrators and application programmers who want to:

- Understand how NonStop SQL/MX differs from NonStop SQL/MP.
- Write new SQL/MX applications.
- Write new SQL/MP applications that will be ported to NonStop SQL/MX in the future.
- Investigate migrating existing SQL/MP applications to NonStop SQL/MX.

Learning about the features of SQL implemented in NonStop SQL/MX enables you, as a database administrator or an application programmer, to choose the appropriate database for your needs. In particular, it is important for those who want to rework SQL/MP applications to NonStop SQL/MX to begin thinking about specific differences.

Related Documentation

This manual is part of the NonStop SQL/MX library of manuals, which includes:

Introductory Guides

SQL/MX Comparison Guide for SQL/MP Users	Describes SQL differences between NonStop SQL/MP and NonStop SQL/MX.
SQL/MX Quick Start	Describes basic techniques for using SQL in the SQL/MX conversational interface (MXCI). Includes information about installing the sample database.

Reference Manuals

SQL/MX Reference Manual	Describes the syntax of SQL/MX statements, MXCI commands, functions, and other SQL/MX language elements.
SQL/MX Connectivity Service Administrative Command Reference	Describes the SQL/MX administrative command library (MACL) available with the SQL/MX conversational interface (MXCI).
DataLoader/MX Reference Manual	Describes the features and functions of the DataLoader/MX product, a tool to load SQL/MX databases.

SQL/MX Messages Manual	Describes SQL/MX messages.
SQL/MX Glossary	Defines SQL/MX terminology.

Programming Manuals

SQL/MX Programming Manual for C and COBOL	Describes how to embed SQL/MX statements in ANSI C and COBOL programs.
SQL/MX Programming Manual for Java	Describes how to embed SQL/MX statements in Java programs according to the SQLJ standard.

Specialized Guides

SQL/MX Installation and Management Guide	Describes how to plan, install, create, and manage an SQL/MX database. Explains how to use installation and management commands and utilities.
SQL/MX Query Guide	Describes how to understand query execution plans and write optimal queries for an SQL/MX database.
SQL/MX Data Mining Guide	Describes the SQL/MX data structures and operations to carry out the knowledge-discovery process.
SQL/MX Queuing and Publish/Subscribe Services	Describes how SQL/MX integrates transactional queuing and publish/subscribe services into its database infrastructure.
SQL/MX Report Writer Guide	Describes how to produce formatted reports using data from a NonStop SQL/MX database.
SQL/MX Connectivity Service Manual	Describes how to install and manage the SQL/MX Connectivity Service (MXCS), which enables applications developed for the Microsoft Open Database Connectivity (ODBC) application programming interface (API) and other connectivity APIs to use SQL/MX.
SQL/MX Guide to Stored Procedures in Java	Describes how to use stored procedures that are written in Java within SQL/MX.

Online Help

The *SQL/MX Online Help* consists of:

Reference Help	Overview and reference entries from the *SQL/MX Reference Manual*.
Messages Help	Individual messages grouped by source from the *SQL/MX Messages Manual.*

Glossary Help	Terms and definitions from the *SQL/MX Glossary*.
NSM/web Help	Context-sensitive help topics that describe how to use the NSM/web management tool.
Visual Query Planner Help	Context-sensitive help topics that describe how to use the Visual Query Planner graphical user interface.

The NSM/web and Visual Query Planner help systems are accessible from their respective applications. You can download the Reference, Messages, and Glossary online help from the $SYSTEM.ZMXHELP subvolume or from the HP NonStop Technical Library (NTL). For more information about downloading online help, see the *SQL/MX Installation and Management Guide*.

These manuals are part of the SQL/MP library of manuals and are essential references for information about SQL/MP Data Definition Language (DDL) and SQL/MP installation and management:

Related SQL/MP Manuals

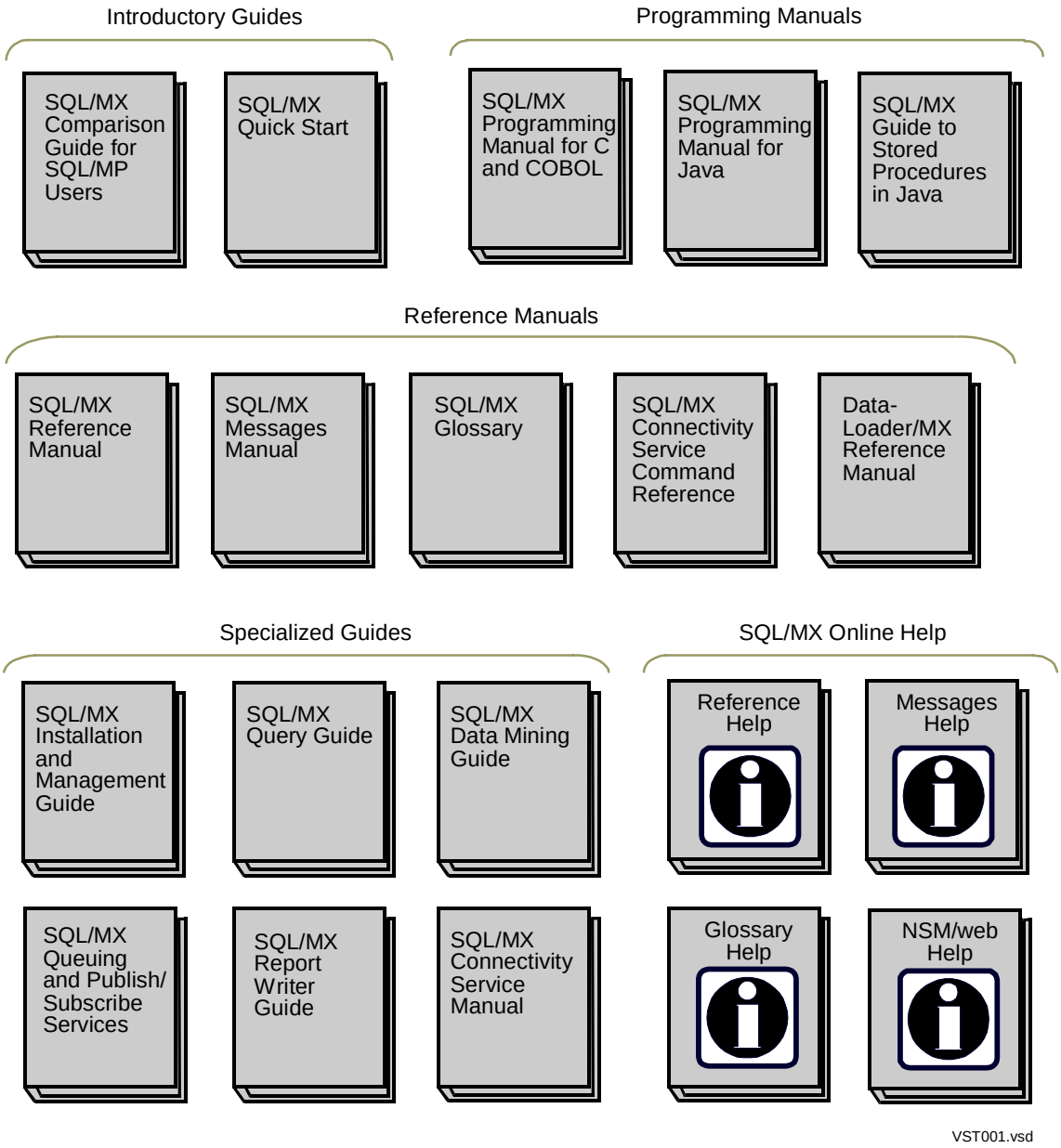
SQL/MP Reference Manual	Describes the SQL/MP language elements, expressions, predicates, functions, and statements.
SQL/MP Installation and Management Guide	Describes how to plan, install, create, and manage an SQL/MP database. Describes installation and management commands and SQL/MP catalogs and files.

This documentation is helpful for understanding the concepts and terminology of this manual:

Other Related Documentation

C/C++ Programmer's Guide	Describes HP extensions to the C and C++ languages for NonStop servers.
SQL/MX Connectivity Service Manual	Describes how to install and manage MXCS, which enables applications developed for Microsoft Open Database Connectivity (ODBC) application programming interface (API) to use SQL/MX.
Extensible Markup Language (XML) Web site	Describes the W3 standards for XML at http://www.w3.org/XML.

This figure shows the manuals in the SQL/MX library:



Notation Conventions

Hypertext Links

Blue underline is used to indicate a hypertext link within text. By clicking a passage of text with a blue underline, you are taken to the location described. For example:

> This requirement is described under Backup DAM Volumes and Physical Disk Drives on page 3-2.

General Syntax Notation

This list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS. Uppercase letters indicate keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
MAXATTACH
```

lowercase italic letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

```
file-name
```

computer type. `Computer type` letters within text indicate C and Open System Services (OSS) keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
myfile.c
```

italic computer type. *`Italic computer type`* letters within text indicate C and Open System Services (OSS) variable items that you supply. Items not enclosed in brackets are required. For example:

```
pathname
```

[ ] Brackets. Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name
```

```
INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
FC [ num  ]
   [ -num ]
   [ text ]
```

```
K [ X | D ] address
```

{ } Braces. A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }
```

```
ALLOWSU { ON | OFF }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis. An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address [ , new-value ]...

[ - ] {0|1|2|3|4|5|6|7|8|9}...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char..."
```

Punctuation. Parentheses, commas, semicolons, and other symbols not previously described must be typed as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;

LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must type as shown. For example:

```
"[" repetition-constant-list "]"
```

Item Spacing. Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In this example, no spaces are permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing. If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] LINE

   [ , attribute-spec ]...
```

1

Comparing the Differences in the Products

NonStop SQL/MX is HP's next-generation relational database management system designed for business-critical applications. SQL/MX software brings traditional NonStop fundamentals—high availability, scalability, reliability, and parallel processing—to a distributed database.

NonStop SQL/MP is HP's relational database management system that was first introduced in 1987. Since then, many new features and performance enhancements have been added. NonStop SQL/MP is widely used today for critical OLTP and DSS applications running on HP NonStop servers.

Why Build a New DBMS?

NonStop SQL/MX builds upon the heritage and the many years of experience gained with NonStop SQL/MP. Building on the capabilities of NonStop SQL/MP, ANSI SQL standard (1992) features have been added that enable you to create applications for open systems. Parallel query capabilities and ease of management have made NonStop SQL/MP the preferred database for large business intelligence and Operational Data Store (ODS) applications. In addition, NonStop SQL/MP has been the solution for business-critical OLTP applications, such as securities trading and stock exchange activities.

NonStop SQL/MP has become the choice for large OLTP, business intelligence, and operational data store applications over the past 12 years. Businesses such as securities trading, stock exchanges, retail, and manufacturing rely on NonStop SQL/MP to manage their daily business, because they require highly available, reliable, and scalable features. Over the years, NonStop SQL/MP has been enhanced with features and is well tuned for both OLTP and DSS environments. HP recognizes the importance of NonStop SQL/MP and will continue to support it.

NonStop SQL/MX uses SQL/MP's capabilities and extends them to include data mining abilities. NonStop SQL/MX inherits those same performance and high-availability characteristics and provides a familiar environment for SQL/MP database professionals by building on the features NonStop SQL/MP currently offers.

So, why this new database management system? NonStop SQL/MP was conceived in 1987, long before the demands of the internet and the needs of e-business. NonStop SQL/MX was created to support the newer ANSI standard, which has evolved considerably since NonStop SQL/MP was first released. Customer needs, relating to scalability, portability, memory, and processor demands, have all contributed to the need to provide a new database. NonStop SQL/MX meets the needs of both the internet and the e-business market.

Frequently Asked Questions About Differences

The following list of questions about the interaction between NonStop SQL/MP and NonStop SQL/MX was generated from customer response at a recent user conference.

- Will customers be forced to migrate applications from NonStop SQL/MP to NonStop SQL/MX?

  No. The data-sharing architecture allows you to build new applications that take advantage of the new SQL/MX features while accessing data maintained in existing SQL/MP applications. You can choose which pieces of existing applications will be coded for NonStop SQL/MX and which pieces will continue to use NonStop SQL/MP. NonStop SQL/MX and NonStop SQL/MP can run concurrently against the same data for both read and write. A few SQL/MP data types are not supported in NonStop SQL/MX. See Section 3, DML Features for a list of data types supported by NonStop SQL/MX.

- Will the new features of NonStop SQL/MX, such as GRANT/REVOKE, be made available in NonStop SQL/MP?

  No. The SQL/MP architecture does not comply with the ANSI-92 standard.

- What are future plans for NonStop SQL/MP? Will SQL/MP be enhanced with new features?

  NonStop SQL/MP was designed in the mid-1980's using the available technology. The SQL standard has evolved since then. Although support will continue for NonStop SQL/MP, most new features will be implemented only in NonStop SQL/MX because the architecture of NonStop SQL/MX facilitates adding those new features.

- Do you need to change your SQL/MP tables to run NonStop SQL/MX?

  In many cases, yes. Some SQL/MP features do not map directly to NonStop SQL/MX. For descriptions of SQL/MP features that do not map to NonStop SQL/MX, see specific entries in the *SQL/MX Reference Manual*.

- Is dynamic SQL easier to code in NonStop SQL/MX than in NonStop SQL/MP?

  Yes. NonStop SQL/MX uses the ANSI model for dynamic SQL. You use SET DESCRIPTOR and GET DESCRIPTOR instead of SQLCA. You use GET DIAGNOSTICS for error handling instead of SQLDA. See Section 4, Embedded SQL, for additional information about dynamic SQL. In addition, see the *SQL/MX Reference Manual* and the *SQL/MX Programming Guide for C and COBOL*.

- Can you use DEFINEs in NonStop SQL/MX?

Yes, but DEFINEs for table naming are not necessary in NonStop SQL/MX. DEFINES can be used only for SQL/MP objects referenced through the SQL/MX application. SQL/MX objects cannot be referenced with DEFINES.

In addition, DEFINEs must be available to NonStop SQL/MX at run time, or you will receive an error. NonStop SQL/MP does not have this requirement.

- Does NonStop SQL/MX support similarity checking and execution-time name resolution?

  Yes. NonStop SQL/MX supports similarity checking and execution-time name resolution with some limitations.

- Does NonStop SQL/MX support Report Writer?

  Yes. In addition, you can use third-party tools for reports. If you need assistance in choosing a tool, contact your HP representative.

- Can NonStop SQL/MX run under both the HP NonStop Kernel Open System Services (OSS) and Guardian Services environments?

  Many key SQL/MX processes run as OSS processes (as do HP NonStop Tuxedo, HP NonStop DOM, and Java processes). Your applications can be run on Guardian with an OSS pass-through command. See the *SQL/MX Programming Guide for C and COBOL* for more information.

- What benefits will NonStop SQL/MX provide for OLTP applications?

  You can now build your OLTP application by using ANSI-standard SQL. This feature allows you to leverage the expertise that your developers have gained in working on other ANSI-standard databases and quickly transfer those skills to SQL/MX development. It also allows you to build applications that can be ported between DBMS systems with a minimum of change. Several new SQL/MX features, including rowsets and compound statements, provide significant performance improvements for OLTP applications. NonStop SQL/MX also supports a publish and subscribe feature that supports transactional-based queuing.

- Does TACL support NonStop SQL/MX?

  Yes. TACL supports SQL/MX objects when the Guardian name of the object is specified. TACL commands do not accept ANSI names as input and do not return ANSI names in their output. These TACL commands support SQL/MX objects:

  - FILEINFO
  - FILENAMES
  - FILES
  - #FILEGETLOCKINFO
  - #FILEINFO
  - #FILENAMES
  - #LOCKINFO
  - #OPENINFO
  - #XFILEINFO

- #XFILENAMES
- #XFILES

In general, the commands work the same and return the same information for SQL/MX objects as they do for non-SQL/MX objects. Exceptions are noted in the *TACL Reference Manual.*

These commands return errors when used with SQL/MX objects and are not supported:

- COPYDUMP
- PURGE
- RENAME
- #PURGE
- #RENAME

What's New in NonStop SQL/MX?

Redesigned Optimizer and Executor

NonStop SQL/MX uses a redesigned optimizer and executor. The top-down, cost-based, rules-driven optimizer considers more plans than NonStop SQL/MP. Extensibility is provided through the compiler, because new transformations using rules can be added with ease to the optimizer. The redesigned executor also provides extensibility that uses a data-flow architecture. The executor takes advantage internally of completely no-waited operations and provides for more work in the HP NonStop Data Access Manager (DAM), including multiple forms of parallelism and scalability on multiple levels.

For additional information about differences between the SQL/MP and SQL/MX optimizer features, see Section 5, Optimizer and Executor.

Performance

In addition to considering many more plans, the SQL/MX optimizer allows you to push down certain plans that involve compound statements or nested joins into the Data Access Manager. You can influence a query plan by rewriting portions of a plan with the CONTROL QUERY SHAPE command. NonStop SQL/MX provides histogram statistics that provide more accurate statistics of the data in the database. These statistics are also used by the optimizer to create efficient query plans.

For additional information about optimizer features, see Section 5, Optimizer and Executor.

Parallel Operations for Scalable High Performance

NonStop SQL/MX can execute large queries that scan large portions of the database in parallel. Although NonStop SQL/MX supports three basic forms of parallelism, partitioned parallelism is the main type of parallelism used. Partitioned parallelism is

the ability to divide the data to be processed into partitions and work on each partition in parallel. In a partitioned parallel plan, multiple operators all work on the same plans, and results are merged by using multiple pipelines. Although NonStop SQL/MP uses some partitioned parallelism to process in parallel, NonStop SQL/MX has been designed to take advantage of both executor server process (ESP) parallelism and Data Access Manager parallelism.

For additional information about parallelism, see the *SQL/MX Query Guide*.

Manageability

For parallelism in NonStop SQL/MP, one ESP per partition is started. NonStop SQL/MX simplifies ESP management by starting only the number of ESPs it needs to process a plan in parallel if the optimizer determines that ESP partitioned parallelism will provide the best priced query plan.

NonStop SQL/MX provides a number of user-controlled defaults that allow you to change default settings that affect optimization, performance, and the generation of histogram statistics.

Like NonStop SQL/MP, NonStop SQL/MX provides EXPLAIN output. However, NonStop SQL/MX provides EXPLAIN as a table, and the output is in machine-readable format. You can easily extract the EXPLAIN output from embedded SQL programs. In addition, NonStop SQL/MX provides a GUI-based query tool called Visual Query Planner that graphically displays the results of query plans.

ANSI Compatibility

NonStop SQL/MX is open, portable, and, most important, uses an industry standard: the American National Standards Institute (ANSI) version of SQL that was created in 1992. This version of SQL is referred to as SQL-92—a standard that is considered far more complete than earlier versions of SQL for business-critical applications. SQL-92 was replaced by SQL:1999. Core SQL:1999 contains all of Entry SQL-92, plus much of Intermediate SQL-92 and some of Full SQL-92, plus a few new SQL:1999 features.

NonStop SQL/MX is based on SQL:1999 and includes support for features from higher levels of the standard. NonStop SQL/MP is based on the SQL-89 standard.

Applications with SQL statements that adhere to the SQL:1999 standard can be ported to run against a variety of database management systems, including NonStop SQL/MX. If you are trained to write applications with other SQL-92 compliant database management systems, you can easily apply your skills to applications for NonStop SQL/MX.

For details about how NonStop SQL/MX features comply with ANSI standards, see the *SQL/MX Reference Manual*.

Other Features

Many other features have been added to NonStop SQL/MX that are not discussed in this guide. Some of those features include:

- Data mining
- Publish and subscribe
- Compound statements and rowsets

For additional information regarding these topics, see the *SQL/MX Data Mining Guide*, the *SQL/MX Queuing and Publish/Subscribe Services*, and the *SQL/MX Programming Guide for C and COBOL*.

2 Quick Reference to Differences

Table 2-1 lists the content of NonStop SQL/MP as found in the *SQL/MP Reference Manual*, the *SQL/MX Reference Manual*, and other manuals. The column **Implemented In NonStop SQL/MX?** indicates whether the item is supported in NonStop SQL/MX (yes, no, or changed.) “Changed” means that the item is supported differently in NonStop SQL/MX than it was in NonStop SQL/MP.

NonStop SQL/MX supports Report Writer. In the table, Report Writer commands are identified in the **Comments** column.

Remember that NonStop SQL/MX has many features that are not listed here because this list is NonStop SQL/MP-focused. See the NonStop SQL/MX manuals, especially the *SQL/MX Reference Manual*, for further information.

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 1 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
Access option	Changed	See Section 3, DML Features
ADD DEFINE	Yes	See *SQL/MX Reference Manual*
Aggregate function	Yes	See Section 3, DML Features and the *SQL/MX Reference Manual*
Alias	Changed	Used differently in each product. See *SQL/MP Reference Manual* and *SQL/MX Reference Manual*
ALLOCATE file attribute	Yes	See *SQL/MX Reference Manual*
ALTER CATALOG statement	No	
ALTER COLLATION statement	No	
ALTER DEFINE statement	Yes	See *SQL/MX Reference Manual*
ALTER INDEX statement	Yes	Fewer capabilities. See *SQL/MX Reference Manual*
ALTER PROGRAM statement	No	
ALTER TABLE statement	Yes	Fewer capabilities. See *SQL/MX Reference Manual*
ALTER VIEW statement	No	
APPEND command	No	
APPENDCANCEL command	No	

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 2 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
APPENDRESTART command	No	
ASCII character set	Yes	
AS clause	Yes	See *SQL/MX Report Writer Guide*
AS DATE/TIME clause	Yes	See *SQL/MX Report Writer Guide*
AUDIT file attribute	No	See *SQL/MP Reference Manual*
AUDITCOMPRESS file attribute	Yes	See *SQL/MX Reference Manual*
Audited tables	Yes	See *SQL/MX Reference Manual*
AVG function	Yes	See *SQL/MX Reference Manual*
BACKUP utility	Yes	See *SQL/MP Reference Manual*
BASETABS table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
BEGIN DECLARE SECTION statement	Yes	See Section 4, Embedded SQL and *SQL/MX Reference Manual*
BEGIN WORK statement	Yes	See Section 3, DML Features and *SQL/MX Reference Manual*
BETWEEN predicate	Yes	See *SQL/MX Reference Manual*
BLOCKSIZE file attribute	Yes	Fewer capabilities. See *SQL/MX Reference Manual*
BREAK FOOTING command	Yes	See *SQL/MX Report Writer Guide*
BREAK ON command	Yes	See *SQL/MX Report Writer Guide*
BREAK TITLE command	Yes	See *SQL/MX Report Writer Guide*
BUFFERED file attribute	No	See *SQL/MP Reference Manual*
CANCEL command	Yes	See *SQL/MX Report Writer Guide*
CASE expression	Yes	See *SQL/MX Reference Manual*
CAST function	Yes	See Section 3, DML Features and *SQL/MX Reference Manual*
CATALOG command	No	See *SQL/MP Reference Manual*
Catalogs	Changed	See Organization of Catalogs and Schemas *on page 6-2*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 3 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
CATALOGS table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
CENTER_REPORT option	Yes	See *SQL/MX Report Writer Guide*
Character data type	Yes	See *SQL/MX Reference Manual*
Character expression	Yes	See *SQL/MX Reference Manual*
Character sets	Yes	See *SQL/MX Reference Manual*
CHAR_LENGTH function	Yes	See *SQL/MX Reference Manual*
CLEANUP command	No	
CLEARONPURGE file attribute	Yes	See *SQL/MX Reference Manual*
CLOSE statement	Yes	See *SQL/MX Reference Manual*
Clustering keys	Yes	See *SQL/MX Reference Manual*
COLLATE clause	Changed	Can only be specified as COLLATE DEFAULT. See *SQL/MX Reference Manual*
Collation definitions	No	
Collations	Yes	Fewer capabilities. See *SQL/MX Reference Manual*
Column Identifier	No	See *SQL/MP Reference Manual*
Columns	Yes	See *SQL/MX Reference Manual*
COLUMNS table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
COMMENT statement	No	
Comments	Yes	See *SQL/MX Reference Manual*
COMMENTS table	No	See *SQL/MP Reference Manual*
COMMIT option	No	
COMMIT WORK statement	Yes	See *SQL/MX Reference Manual*
Comparison predicate	Yes	See *SQL/MX Reference Manual*
COMPUTE_TIMESTAMP function	No	
CONCAT clause	Yes	See *SQL/MX Report Writer Guide*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 4 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
Concurrency	Yes	See *SQL/MX Reference Manual*
Constraints	Yes	In CREATE TABLE and ALTER TABLE statements. See *SQL/MX Reference Manual*
CONSTRNT table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
CONTINUE statement	No	
CONTROL EXECUTOR directive	Changed	Parallelism enabled through SYSTEM_DEFAULTS table. See Section 5, Optimizer and Executor
CONTROL QUERY directive	Changed	Implemented as CONTROL QUERY DEFAULT. See Section 3, DML Features and *SQL/MX Reference Manual*
CONTROL TABLE directive	Changed	See Section 3, DML Features and *SQL/MX Reference Manual*
CONVERT command	No	See *SQL/MP Reference Manual*
CONVERTTIMESTAMP function	Yes	See *SQL/MX Reference Manual*
COPY command	No	See *SQL/MP Reference Manual*
Correlation names	Yes	See *SQL/MX Reference Manual*
COUNT function	Yes	See *SQL/MX Reference Manual*
CPRLSRCE table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
CPRULES table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
CREATE CATALOG statement	Changed	No security capabilities. See *SQL/MX Reference Manual*
CREATE COLLATION statement	No	See *SQL/MP Reference Manual*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 5 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
CREATE CONSTRAINT statement	Changed	Performed with CREATE TABLE statement. See *SQL/MX Reference Manual*
CREATE INDEX statement	Yes	Some differences. See *SQL/MX Reference Manual*
CREATE SYSTEM CATALOG statement	Changed	See Organization of Catalogs and Schemas *on page 6-2* and *SQL/MX Reference Manual*
CREATE TABLE statement	Yes	Some differences. See *SQL/MX Reference Manual*
CREATE VIEW statement	Changed	Some differences. See Views on page 6-6 and *SQL/MX Reference Manual*
CURRENT function	Yes	See *SQL/MX Reference Manual*
CURRENT_TIMESTAMP function	Yes	See *SQL/MX Reference Manual*
Cursors	Yes	See *SQL/MX Programming Guide for C and COBOL* and *SQL/MX Reference Manual*
c89	Yes	See *SQL/MX Programming Guide for C and COBOL*
Data Dictionary	Yes	Metadata in NonStop SQL/MX. See *SQL/MX Reference Manual*
Data types	Yes	See *SQL/MX Reference Manual*
DATE data type	Yes	See *SQL/MX Reference Manual*
DATE_FORMAT option	Yes	See *SQL/MX Report Writer Guide*
Date-time data types	Changed	See Section 3, DML Features
Date-time functions	Changed	See Section 3, DML Features
Date-time literals	Changed	See Section 3, DML Features
DATEFORMAT function	Yes	See *SQL/MX Reference Manual*
DATETIME data type	Changed	See Section 3, DML Features and *SQL/MX Reference Manual*
DAYOFWEEK function	Yes	See *SQL/MX Reference Manual*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 6 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
DCL (Data Control Language) statements	Changed	See *SQL/MX Reference Manual*
Deadlocks	Yes	See Section 3, DML Features and *SQL/MX Reference Manual* (CONTROL TABLE)
DEALLOCATE file attribute	Yes	See *SQL/MX Reference Manual*
DECIMAL_POINT option	Yes	See *SQL/MX Report Writer Guide*
DECLARE CURSOR statement	Yes	See *SQL/MX Programming Guide for C and COBOL* and *SQL/MX Reference Manual*
DCOMPRESS	No	See *SQL/MP Reference Manual*
DEFAULT clause	Yes	See *SQL/MX Reference Manual*
DEFINEs	Yes	See *SQL/MX Reference Manual*
DELETE DEFINE statement	Yes	See *SQL/MX Reference Manual*
DELETE statement	Yes	Some differences. See *SQL/MX Reference Manual* and see *SQL/MX Programming Guide for C and COBOL*
DESCRIBE INPUT statement	Yes	See *SQL/MX Programming Guide for C and COBOL* and *SQL/MX Reference Manual*
DESCRIBE statement	Yes	See *SQL/MX Reference Manual*
Detail alias	No	
DETAIL command	Yes	See *SQL/MX Report Writer Guide*
DISPLAY STATISTICS command	Changed	See *SQL/MX Reference Manual*
DISPLAY USE OF command	Changed	Differently implemented. See DISPLAY USE OF Command on page 7-1, Some functions replaced by `mxtool` INFO command
DISTINCT clause	Yes	See *SQL/MX Reference Manual*
Data Manipulation Language (DML) statements	Yes	See Section 3, DML Features and *SQL/MX Reference Manual*
DOWNGRADE CATALOG command	No	See *SQL/MP Reference Manual*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 7 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
DOWNGRADE SYSTEM CATALOG command	No	See *SQL/MP Reference Manual*
DROP statement	Changed	Replaced by specific DROP *object* statements. See *SQL/MX Reference Manual*
DROP SYSTEM CATALOG statement	Changed	Replaced by DROP SQL statement. See *SQL/MX Reference Manual*
DSL (Data Status Language) statements	No	
DSLACK file attribute	No	See *SQL/MP Reference Manual*
DUP command	Yes	See *SQL/MX Reference Manual*
Dynamic SQL	Yes	See Section 4, Embedded SQL and *SQL/MX Programming Guide for C and COBOL*
EDIT command	No	
Embedded SQL	Yes	See Section 4, Embedded SQL, *SQL/MX Reference Manual*, *SQL/MX Programming Guide for C and COBOL,* and *SQL/MX Programming Guide for C and COBOL*.
END DECLARE SECTION statement	Yes	See *SQL/MX Reference Manual*
ENV command	Yes	See *SQL/MX Reference Manual*
ERROR command	Yes	See *SQL/MX Reference Manual*
Error messages	Yes	See *SQL/MX Reference Manual*
EXECUTE statement	Yes	See *SQL/MX Reference Manual*
EXECUTE IMMEDIATE statement	Yes	See *SQL/MX Reference Manual*
EXISTS predicate	Yes	See *SQL/MX Reference Manual*
EXIT command	Yes	See *SQL/MX Reference Manual*
EXPLAIN directive	Changed	See Section 5, Optimizer and Executor, *SQL/MX Query Guide,* and *SQL/MX Reference Manual*
Expressions	Yes	See *SQL/MX Reference Manual*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 8 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
EXTEND function	No	
EXTENT file attribute	Yes	See *SQL/MX Reference Manual*
FC command	Yes	See *SQL/MX Reference Manual*
FETCH statement	Yes	See *SQL/MX Reference Manual*
File attributes	Changed	See *SQL/MP Reference Manual*
File organization	Changed	See *SQL/MP Reference Manual*
FILEINFO command	Changed	Replaced SHOWLABEL utility. See *SQL/MX Reference Manual*.
FILENAMES command	No	
FILES command	No	
FILES table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
Filesets	No	
Floating point data type for C and C++	Yes	See the *SQL/MX Reference Manual*
FREE RESOURCES statement	No	
Functions	Yes	See Section 3, DML Features and *SQL/MX Reference Manual*
FUP command	No	
Generalized owner	No	
GET CATALOG OF SYSTEM statement	No	
GET VERSION OF PROGRAM statement	No	
GET VERSION statement	No	
GOAWAY command	Changed	Replaced by `mxtool` FIXUP command
Group manager	No	
Guardian names	Yes	See the *SQL/MX Reference Manual*
HEADING clause	Yes	See *SQL/MX Reference Manual*
HEADINGS option	Yes	See *SQL/MX Report Writer Guide*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 9 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
HELP command	Changed	HTML Help will be provided
HELP TEXT statement	No	
HISTORY command	Yes	See *SQL/MX Reference Manual*
Host identifiers	Yes	See *SQL/MX Programming Guide for C and COBOL*
Host programs	Yes	See *SQL/MX Programming Guide for C and COBOL*
Host variables	Yes	See Section 4, Embedded SQL and *SQL/MX Programming Guide for C and COBOL*
ICOMPRESS file attribute	No	
IF/THEN/ELSE clause	Yes	See *SQL/MX Report Writer Guide*
IN predicate	Yes	See *SQL/MX Reference Manual*
INCLUDE SQLCA directive	Changed	Use diagnostics area for error handling. See Section 4, Embedded SQL and *SQL/MX Programming Guide for C and COBOL*
INCLUDE SQLDA directive	Changed	Use descriptor area for error handling. See Section 4, Embedded SQL and *SQL/MX Programming Guide for C and COBOL*
INCLUDE SQLSA directive	No	
INCLUDE STRUCTURES directive	No	
Index keys	Yes	See *SQL/MX Reference Manual*
INDEXES table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
Indicator variables and Indicator parameters	Yes	See Section 4, Embedded SQL and *SQL/MX Programming Guide for C and COBOL*
INFO DEFINE statement	Yes	See *SQL/MX Reference Manual*
INITIALIZE SQL command	Yes	See *SQL/MX Reference Manual*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 10 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
INSERT statement	Yes	See *SQL/MX Reference Manual* and *SQL/MX Programming Guide for C and COBOL*
INTERVAL data type	Yes	See *SQL/MX Reference Manual*
INTERVAL literal	Yes	See *SQL/MX Reference Manual*
INVOKE directive and command	Changed	Some functions replaced by SQL/MX SHOWDDL command. See Section 4, Embedded SQL, *SQL/MX Programming Guide for C and COBOL* and *SQL/MX Reference Manual*
ISLACK file attribute	No	
Joins	Yes	See *SQL/MX Reference Manual*
JULIANTIMESTAMP function	Yes	See *SQL/MX Reference Manual*
Keys	Yes	See *SQL/MX Reference Manual*
KEYS table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
LEFT_MARGIN option	Yes	See *SQL/MX Report Writer Guide*
LIKE predicate	Yes	See *SQL/MX Reference Manual*
Limits	Yes	See *SQL/MX Reference Manual*
LINE_NUMBER function	Yes	See *SQL/MX Report Writer Guide*
LINE_SPACING option	Yes	See *SQL/MX Report Writer Guide*
LIST command	Yes	See *SQL/MX Report Writer Guide*
Literals	Yes	See Section 3, DML Features and *SQL/MX Reference Manual*
LOAD command	Changed	Replaced by IMPORT utility. See *SQL/MX Reference Manual*
LOCK TABLE statement	Yes	See *SQL/MX Reference Manual*
Locking	Yes	See *SQL/MX Reference Manual*
LOCKLENGTH file attribute	No	
LOG command	Yes	See *SQL/MX Reference Manual*
LOGICAL_FOLDING option	Yes	See *SQL/MX Report Writer Guide*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 11 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
MAX function	Yes	See *SQL/MX Reference Manual*
MAXEXTENTS file attribute	Yes	See *SQL/MX Reference Manual*
Message file	Changed	See *SQL/MX Online Help* (HTML format) for messages and online help text
MIN function	Yes	See *SQL/MX Reference Manual*
MODIFY CATALOG command	No	
MODIFY LABEL command	No	
MODIFY REGISTER command	No	
Multibyte character sets	Yes	
NAME command	Yes	See *SQL/MX Report Writer Guide*
NAME option	No	
Name resolution	Yes	See *SQL/MX Reference Manual* and *SQL/MX Programming Guide for C and COBOL*
Names	Yes	See *SQL/MX Reference Manual*
NEWLINE_CHAR option	Yes	See *SQL/MX Report Writer Guide*
Nonaudited tables	No	
NOPURGEUNTIL file attribute	No	
NULL predicate	Yes	See *SQL/MX Reference Manual*
Null values	Yes	See *SQL/MX Reference Manual*
NULL_DISPLAY	Yes	See *SQL/MX Report Writer Guide*
Numeric data types	Yes	See *SQL/MX Reference Manual*
Numeric literals	Yes	See *SQL/MX Reference Manual*
OBEY command	Yes	See *SQL/MX Reference Manual*
OCTET_LENGTH function	Yes	See *SQL/MX Reference Manual*
OPEN statement	Yes	See *SQL/MX Reference Manual*
OSS names	Yes	
OUT command	No	

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 12 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
OUT_REPORT command	Yes	See *SQL/MX Report Writer Guide*
OVERFLOW_CHAR command	Yes	See *SQL/MX Report Writer Guide*
OWNER file attribute	No	
PAGE_COUNT option	Yes	See *SQL/MX Report Writer Guide*
PAGE_FOOTING option	Yes	See *SQL/MX Report Writer Guide*
PAGE_LENGTH option	Yes	See *SQL/MX Report Writer Guide*
PAGE_NUMBER option	Yes	See *SQL/MX Report Writer Guide*
PAGE TITLE option	Yes	See *SQL/MX Report Writer Guide*
Parallel index loading	Changed	NonStop SQL/MX does parallel index loading by default. It is not a specifiable attribute.
Parameters	Yes	No TYPE AS clause. See *SQL/MX Reference Manual*
PARTITION clause	Yes	See *SQL/MX Reference Manual*
Partitions	Yes	See *SQL/MX Reference Manual*
Partition management	Yes	Performed by SQL/MX MODIFY command. See *SQL/MX Reference Manual*
PARTNS table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
PERUSE command	No	
Plans	Yes	See *SQL/MX Query Guide*
POSITION function	Yes	See *SQL/MX Reference Manual*
Predicates	Yes	See Section 3, DML Features and *SQL/MX Reference Manual*
PREPARE statement	Yes	See *SQL/MX Reference Manual* and *SQL/MX Programming Guide for C and COBOL*
Primary keys	Yes	See *SQL/MX Reference Manual*
Print item	No	
PROGID file attribute	No	

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 13 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
Program invalidation	No	
PROGRAMS table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
Protection view	No	See Views on page 6-6
PURGE command	No	
PURGEDATA command	Yes	See *SQL/MX Reference Manual*
Qualified fileset list	No	
Quantified predicate	Yes	See *SQL/MX Reference Manual*
RECLENGTH file attribute	No	
RELEASE statement	No	
REPORT FOOTING command	Yes	See *SQL/MX Report Writer Guide*
REPORT option	No	
REPORT TITLE command	Yes	See *SQL/MX Report Writer Guide*
Reserved words	Yes	See *SQL/MX Reference Manual*
RESETBROKEN file attribute	Changed	Replaced by `mxtool` FIXUP command
RESET DEFINE statement	No	See *SQL/MX Reference Manual*
RESET LAYOUT command	Yes	See *SQL/MX Report Writer Guide*
RESET PARAM command	Yes	See *SQL/MX Reference Manual*
RESET PREPARED command	No	
RESET REPORT command	Yes	See *SQL/MX Report Writer Guide*
RESET SESSION command	Yes	See *SQL/MX Report Writer Guide*
RESET STYLE command	Yes	See *SQL/MX Report Writer Guide*
RESETBROKEN file attribute	No	
RIGHT_MARGIN option	Yes	See *SQL/MX Report Writer Guide*
ROLLBACK WORK statement	Yes	See *SQL/MX Reference Manual*
ROWCOUNT option	Yes	See *SQL/MX Report Writer Guide*
Sample database	Yes	See *SQL/MX Reference Manual*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 14 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
SAVE command	No	
Search conditions	Yes	See *SQL/MX Reference Manual*
SECURE command	No	Replaced by GRANT and REVOKE statements. See *SQL/MX Reference Manual*
SECURE file attribute	No	
Security	Yes	See *SQL/MP Reference Manual*
SELECT statement	Yes	Some differences. See *SQL/MX Reference Manual*
SERIALWRITES file attribute	No	
SET DEFINE statement	Yes	See *SQL/MX Reference Manual*
SET DEFMODE command	No	
SET LAYOUT command	Yes	See *SQL/MX Report Writer Guide*
SET PARAM command	Yes	See *SQL/MX Reference Manual* See *SQL/MX Report Writer Guide*
SET SESSION command	No	
SET STYLE command	Yes	See *SQL/MX Report Writer Guide*
SETSCALE function	Changed	See Section 4, Embedded SQL
Shorthand view	No	See Views on page 6-6
SHOW CONTROL command	Yes	See *SQL/MX Reference Manual*
SHOW DEFINE statement	Yes	See *SQL/MX Reference Manual*
SHOW DEFMODE command	No	
SHOW LAYOUT command	Yes	See *SQL/MX Report Writer Guide*
SHOW PARAM command	Yes	See *SQL/MX Reference Manual*
SHOW PREPARED command	No	
SHOW REPORT command	Yes	See *SQL/MX Report Writer Guide*
SHOW SESSION command	Yes	See *SQL/MX Reference Manual* See *SQL/MX Report Writer Guide*
SHOW STYLE command	Yes	*SQL/MX Report Writer Guide*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 15 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
Similarity checks	Changed	Handled internally by NonStop SQL/MX. See Similarity Checks and Automatic Recompilation on page 4-13
SLACK file attribute	No	
SMF	No	SQL/MX objects cannot be on SMF-managed volumes.
SPACE option	Yes	See *SQL/MX Report Writer Guide*
SQL directive	Yes	See *SQL/MX Programming Guide for C and COBOL*
SQL identifiers	Yes	See *SQL/MX Reference Manual*
SQLDA area	Changed	See Dynamic SQL on page 4-6
SQLCI	Changed	MXCI in NonStop SQL/MX. See *SQL/MX Reference Manual* for MXCI command. See *SQL/MP Reference Manual* for SQLCI command.
SQLCI commands	Changed	MXCI in NonStop SQL/MX. See MXCI Commands in *SQL/MX Reference Manual*
SQLCODE	Changed	See *SQL/MX Programming Guide for C and COBOL*
SQLCOMP command	Changed	MXCI in NonStop SQL/MX. See *SQL/MX Programming Guide for C and COBOL*
SQLSA area	No	See Statistics Area on page 4-10
Standards conformance	Yes	See Section 1, Comparing the Differences in the Products and *SQL/MX Reference Manual*
Statements	Yes	Some differences. See SQL DML Statements on page 3-12 and *SQL/MX Reference Manual*
Static SQL	Yes	See *SQL/MX Programming Guide for C and COBOL*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 16 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
Statistics	Changed	See Section 5, Optimizer and Executor and *SQL/MX Reference Manual*
HP NonStop Storage Management Foundation (SMF)	No	SQL/MX objects cannot be on SMF-managed volumes.
Stored procedures	Yes	See *SQL/MX Guide to Stored Procedures in Java*
String functions	Yes	See Section 3, DML Features and *SQL/MX Reference Manual*
String literals	Yes	See *SQL/MX Reference Manual*
Subqueries	Yes	See *SQL/MX Reference Manual*
SUBSTRING function	Yes	See *SQL/MX Reference Manual*
SUBTOTAL command	Yes	See *SQL/MX Report Writer Guide*
SUBTOTAL_LABEL option	Yes	See *SQL/MX Report Writer Guide*
SUM function	Yes	See *SQL/MX Reference Manual*
Super ID	Yes	See Authorization ID in *SQL/MX Reference Manual*
SYSKEY	Yes	See *SQL/MX Reference Manual*
System catalog	Changed	Known as Metadata in NonStop SQL/MX. See *SQL/MX Reference Manual*
SYSTEM command	No	
System DEFINEs	Yes	See *SQL/MX Reference Manual*
TABLECODE file attribute	No	See *SQL/MP Reference Manual*
Tables	Yes	See *SQL/MX Reference Manual*
TABLES table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
TEDIT command	No	
Temporary tables	Yes	Used internally only in NonStop SQL/MX.
TIME_FORMAT option	Yes	See *SQL/MX Report Writer Guide*

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 17 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
TIME data type	Yes	See Section 3, DML Features and *SQL/MX Reference Manual*
TIMESTAMP data type	Yes	See Section 3, DML Features and *SQL/MX Reference Manual*
TMF transactions	Yes	See *Transaction Management Facility (TMF) Introduction*
TOTAL command	No	
TRANSID table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
TRIM function	Yes	See *SQL/MX Reference Manual*
UNDERLINE_CHAR option	Yes	See *SQL/MX Report Writer Guide*
UNLOCK TABLE statement	Yes	See *SQL/MX Reference Manual*
UPDATE statement	Yes	See *SQL/MX Reference Manual* and *SQL/MX Programming Guide for C and COBOL*
UPDATE STATISTICS statement	Changed	See Section 5, Optimizer and Executor, *SQL/MX Query Guide*, and *SQL/MX Reference Manual*
UPGRADE CATALOG command	No	
UPGRADE SYSTEM CATALOG command	No	
UPSHIFT function	Yes	See *SQL/MX Reference Manual*
USAGES table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
User-defined keys	Yes	See *SQL/MX Reference Manual*
Utilities	Changed	See Section 7, Utility Differences
VARCHAR_WIDTH option	Yes	See *SQL/MX Report Writer Guide*
VERIFIEDWRITES file attribute	No	
VERIFY command	Yes	Replaced by `mxtool` VERIFY command

Table 2-1. NonStop SQL/MP Contents Compared to NonStop SQL/MX (page 18 of 18)

NonStop SQL/MP Item	Implemented in NonStop SQL/MX?	Comments
Versions	No	
VERSIONS table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
Views	Changed	No protection or shorthand views. See Views on page 6-6.
VIEWS table	Changed	Metadata tables are implemented differently. See *SQL/MX Reference Manual*
VOLUME command	No	
WHENEVER directive	Yes	See Section 4, Embedded SQL and *SQL/MX Programming Guide for C and COBOL*
WHERE clause	Yes	See *SQL/MX Reference Manual*
WINDOW option	Yes	See *SQL/MX Report Writer Guide*
WITH SHARED ACCESS option	No	
! command	Yes	

3 DML Features

NonStop SQL/MX contains many SQL/MP extensions to the ANSI standard to enable users to make a smooth transition to NonStop SQL/MX. See “Using NonStop SQL/MX to Access SQL/MP Databases” in the *SQL/MX Reference Manual* if you have questions about how to use SQL/MP tables with the SQL/MX compiler and executor. Also, see the specific information you need to use date-time data types and literals, INTERVAL data types and literals, and CHARACTER string data types and literals.

Although the current release of NonStop SQL/MP conforms to the SQL-89 standard, some major differences exist between NonStop SQL/MP and SQL:1999 in the areas of data types, expressions and functions, DML statement syntax, access options, and transaction management. These areas also show up as differences between NonStop SQL/MP and NonStop SQL/MX that are discussed in this section.

Data Types

NonStop SQL/MX supports most SQL/MP data types. Some SQL/MP DATETIME data types are mapped to the SQL/MX SQL:1999 date-time data types DATE, TIME, and TIMESTAMP. NonStop SQL/MX recognizes the nonstandard FRACTION keyword and maps it directly to the fractional precision of the corresponding SQL:1999 compliant data type.

Date-time Data Types

Both NonStop SQL/MP and NonStop SQL/MX support the SQL:1999 date-time data types of DATE, TIME, and TIMESTAMP. Both NonStop SQL/MP and NonStop SQL/MX support a number of nonstandard date-time types with specific start-end fields. However, NonStop SQL/MX maps some date-time data types to DATE, TIME and TIMESTAMP data types. NonStop SQL/MX does not support nonstandard date-time data types with a start field of FRACTION.

Table 3-1 lists the mapping between date-time data types used in NonStop SQL/MP and data types supported in NonStop SQL/MX. For information about how NonStop SQL/MX processes nonstandard date-time data types, see the *SQL/MX Reference Manual*.

Selecting Supported DATETIME Columns

NonStop SQL/MX supports accessing any SQL/MP DATETIME column except those consisting of FRACTION only.

Suppose that an SQL/MP table has a DATETIME column defined as:

```
MPDateTimeCol DATETIME MONTH TO DAY
DEFAULT DATETIME '03-12' MONTH TO DAY
```

In NonStop SQL/MX you can select from this SQL/MP column as follows:

```
SELECT MPDateTimeCol FROM MPTable;

MPDateTimeCol
-------------
...
03-12
...
```

Selecting FRACTION-Only DATETIME Columns

If you attempt to select data from a FRACTION-only DATETIME column, the value is returned as a string of '#' characters with the same display length as the length of the column.

Suppose that an SQL/MP table has a DATETIME column defined as:

```
MPDateTimeCol DATETIME FRACTION(6)
DEFAULT DATETIME '123456' FRACTION(6)
```

You cannot select the data from this column. For example:

```
SELECT MPDateTimeCol FROM MPTable;
MPDateTimeCol
-------------
######
######

...
```

NonStop SQL/MX returns a warning indicating that you selected an unsupported data type with undefined contents. However, if you try to use the unsupported data type in any sort of expression, NonStop SQL/MX returns an error.

The SQL/MP DATETIME columns that do not map to standard SQL/MX types are represented as SQL/MP DATETIME types in NonStop SQL/MX. These types are listed in Table 3-1.

In embedded SQL, the method for declaring input and output host variables with date-time data types for NonStop SQL/MX is different from the SQL/MP method. See Section 4, Embedded SQL for more information about host variables.

Interval Data Type

NonStop SQL/MX supports the SQL:1999 INTERVAL data type. NonStop SQL/MX recognizes the nonstandard FRACTION keyword used in NonStop SQL/MP and maps it to the equivalent fractional portion of the standard SECOND field. However, it does not support interval data types with a start field of FRACTION.

This table lists the mapping between the interval data types used in NonStop SQL/MP and those used in NonStop SQL/MX:

NonStop SQL/MP Interval Type	Mapping in NonStop SQL/MX
INTERVAL DAY(n) TO SECOND	INTERVAL DAY(n) TO SECOND(0)
INTERVAL DAY(n) TO FRACTION(f)	INTERVAL DAY(n) TO SECOND(f)
INTERVAL HOUR(n) TO SECOND	INTERVAL HOUR(n) TO SECOND(0)
INTERVAL HOUR(n) TO FRACTION(f)	INTERVAL HOUR(n) TO SECOND(f)
INTERVAL MINUTE(n) TO SECOND	INTERVAL MINUTE(n) TO SECOND(0)
INTERVAL MINUTE(n) TO FRACTION(f)	INTERVAL MINUTE(n) TO SECOND(f)
INTERVAL SECOND(n) TO SECOND	INTERVAL SECOND(n,0)
INTERVAL SECOND(n) TO FRACTION(f)	INTERVAL SECOND(n,f)
INTERVAL FRACTION(n) TO FRACTION(f)	Not supported

NonStop SQL/MX handles the interval data types in the next list in the same manner as NonStop SQL/MP:

- YEAR TO (YEAR, MONTH)
- MONTH TO MONTH
- DAY TO (DAY, HOUR, MINUTE)
- HOUR TO (HOUR, MINUTE)
- MINUTE TO MINUTE

For precision and default of interval data types, NonStop SQL/MP and NonStop SQL/MX are the same. For additional information about interval data types, see the *SQL/MX Reference Manual*.

Selecting FRACTION-Only Interval Columns

If you attempt to select data from a FRACTION-only interval column, the value is returned as a string of '#' characters with the same display length as the length of the column.

Suppose that an SQL/MP table has an INTERVAL column defined as:

```
MPIntervalCol INTERVAL FRACTION(6)
DEFAULT INTERVAL '123456' FRACTION(6)
```

You cannot select the data from this column. For example, if you enter this statement:

```
SELECT MPIntervalCol FROM MPTable;
MPIntervalCol
-------------
######
######
...
```

NonStop SQL/MX returns a warning indicating that you selected an unsupported data type with undefined contents. If you try to use the unsupported data type in any sort of expression, NonStop SQL/MX returns an error.

Floating-Point Data Types

NonStop SQL/MP supports floating-point data types of FLOAT, REAL, and DOUBLE PRECISION. A column in an SQL/MP table declared with a floating-point data type is stored in Tandem floating-point format, and all computations on it are done assuming that.

In previous releases of NonStop SQL/MX, if you want an application to input or output a date-time or interval value, you must cast it to STRING data type and then input or output it to or from NonStop SQL/MP.This makes the use of date-time or interval data types cumbersome in application programs and is not ANSI compliant.

SQL/MX tables now store columns in IEEE floating-point format, as data types of FLOAT, REAL, and DOUBLE PRECISION. This allows input and output of data from and to application host variables in Tandem or IEEE floating-point.

NonStop SQL/MX can select data from both SQL/MX and SQL/MP tables. The returned data from mxci is always in IEEE format. The returned data in an embedded program can be in IEEE or Tandem format.

NonStop SQL/MX does not allow any column of the partitioning key of a partitionable table or index to be float data type. If there is no partitioning key specified, the clustering key becomes the partitioning key.

For information about how SQL/MX floating-point data types, see “Numeric Data Types” in the *SQL/MX Reference Manual*.

File Types

SQL/MP tables can have one of three physical file organizations: key-sequenced, entry-sequenced, or relative and can be partitioned. You can access these types of SQL/MP files through NonStop SQL/MX:

- Key-sequenced tables with or without partitions
- Entry-sequenced tables that are not partitioned

You cannot access these types of SQL/MP files through NonStop SQL/MX:

- Entry-sequenced tables that are partitioned
- Relative tables

For more information about SQL/MP file organizations, see the *SQL/MP Reference Manual.*

Identifiers

NonStop SQL/MP supports only regular identifiers, formed from alphanumeric characters and the underscore character. Regular identifiers are not case sensitive. In NonStop SQL/MP, an SQL identifier can contain up to 30 letters (A through Z or a through z), digits (0 through 9), or underscore (_) characters. The first character must be a letter.

NonStop SQL/MX supports both regular and delimited identifiers. An identifier of either type can contain up to 128 characters. Regular identifiers are defined as they are in NonStop SQL/MP—with the exception of length. Delimited identifiers are character strings that appear within double quote characters (") and consist of alphanumeric characters and any character in the SQL-92 special character set except for the at sign (@), the forward slash (/), backward slash (\), and circumflex (^).

For additional information about identifiers, see the *SQL/MX Reference Manual*.

Reserved Words

In NonStop SQL/MP, you should not use reserved words as names of constraints, columns (including correlation names), cursors, or statements. You can, however, use reserved words in catalog names, in Guardian names that identify tables, indexes, views, partitions, collations, and in host variables.

NonStop SQL/MX reserves a longer list of words and allows the use of a reserved word only if you place double quotes around it and observe case-sensitivity. You cannot access views that are defined on tables named after reserved words. See the *SQL/MP Reference Manual* for a list of SQL/MP reserved words. See the *SQL/MX Reference Manual* for a list of SQL/MX reserved words. In addition, see the Identifiers entry in the *SQL/MX Reference Manual* for information about how reserved words can be used and limitations in NonStop SQL/MX.

Literals

NonStop SQL/MX uses single quote characters (') as character string delimiters. NonStop SQL/MX supports all SQL/MP date-time literals. However, if you try to use a literal with unsupported FRACTION-only INTERVAL or DATETIME in any sort of expression, NonStop SQL/MX returns an error.

For additional information about literals, see the *SQL/MX Reference Manual*.

String Delimiters

NonStop SQL/MP uses both single (') and double quote characters (") to enclose character, date-time, and interval literals. NonStop SQL/MX uses only single quote characters for string delimiters, conforming to the SQL:1999 standard. NonStop SQL/MX uses double quotes for delimited identifiers.

Date-time Literals

In NonStop SQL/MP, a date-time literal is a DATETIME, DATE, TIME, or TIMESTAMP literal that you can use in an expression and which can appear in default, USA, or European format.

In NonStop SQL/MX, a date-time literal is a DATE, TIME, or TIMESTAMP literal. An SQL/MX date-time literal begins with the DATE, TIME, or TIMESTAMP keyword and can appear in default, USA, or European format.

For example, these DATE literals are in default, USA, and European formats, respectively:

```
DATE '1990-01-22'
DATE '01/22/1990'
DATE '22.01.1990'
```

Supported and Unsupported SQL/MP Date-Time Literals in NonStop SQL/MX

NonStop SQL/MP date-time literals fall into one of these categories:

- Those that map directly to SQL:1999 types and are supported by NonStop SQL/MX
- Those that do not map to SQL:1999 types but are supported by NonStop SQL/MX
- Those that do not map to SQL:1999 types and are not supported in NonStop SQL/MX

NonStop SQL/MP date-time literals that are not supported are listed in Table 3-1:

Table 3-1. Support For NonStop SQL/MP DATETIME Literals in NonStop SQL/MX (page 1 of 3)

NonStop SQL/MP DATETIME Type	Supported in NonStop SQL/MX	Maps To ANSI Type
DATETIME YEAR TO DAY	Yes	DATE
DATETIME YEAR TO SECOND	Yes	TIMESTAMP(0)
DATETIME YEAR TO FRACTION (1)	Yes	TIMESTAMP(1)
DATETIME YEAR TO FRACTION (2)	Yes	TIMESTAMP(2)
DATETIME YEAR TO FRACTION (3)	Yes	TIMESTAMP(3)
DATETIME YEAR TO FRACTION (4)	Yes	TIMESTAMP(4)
DATETIME YEAR TO FRACTION (5)	Yes	TIMESTAMP(5)
DATETIME YEAR TO FRACTION (6)	Yes	TIMESTAMP(6)
DATETIME HOUR TO SECOND	Yes	TIME(0)
DATETIME HOUR TO FRACTION (1)	Yes	TIME(1)
DATETIME HOUR TO FRACTION (2)	Yes	TIME(2)

Table 3-1. Support For NonStop SQL/MP DATETIME Literals in NonStop SQL/MX (page 2 of 3)

NonStop SQL/MP DATETIME Type	Supported in NonStop SQL/MX	Maps To ANSI Type
DATETIME HOUR TO FRACTION (3)	Yes	TIME(3)
DATETIME HOUR TO FRACTION (4)	Yes	TIME(4)
DATETIME HOUR TO FRACTION (5)	Yes	TIME(5)
DATETIME HOUR TO FRACTION (6)	Yes	TIME(6)
DATETIME YEAR	Yes	(None)
DATETIME YEAR TO MONTH	Yes	(None)
DATETIME YEAR TO HOUR	Yes	(None)
DATETIME YEAR TO MINUTE	Yes	(None)
DATETIME MONTH	Yes	(None)
DATETIME MONTH TO DAY	Yes	(None)
DATETIME MONTH TO HOUR	Yes	(None)
DATETIME MONTH TO MINUTE	Yes	(None)
DATETIME MONTH TO SECOND	Yes	(None)
DATETIME MONTH TO FRACTION(n)	Yes	(None)
DATETIME DAY	Yes	(None)
DATETIME DAY TO HOUR	Yes	(None)
DATETIME DAY TO MINUTE	Yes	(None)
DATETIME DAY TO SECOND	Yes	(None)
DATETIME DAY TO FRACTION(n)	Yes	(None)
DATETIME HOUR	Yes	(None)
DATETIME HOUR TO MINUTE	Yes	(None)
DATETIME HOUR TO SECOND	Yes	(None)
DATETIME HOUR TO FRACTION(n)	Yes	(None)
DATETIME MINUTE	Yes	(None)
DATETIME MINUTE TO SECOND	Yes	(None)
DATETIME MINUTE TO FRACTION(n)	Yes	(None)
DATETIME SECOND	Yes	(None)
DATETIME SECOND TO FRACTION(n)	Yes	(None)
DATETIME FRACTION TO FRACTION(n)	No*	(None)
INTERVAL SECOND(0) TO FRACTION (n)	No*	(None)
INTERVAL FRACTION TO FRACTION (n)	No*	(None)

Table 3-1. Support For NonStop SQL/MP DATETIME Literals in NonStop SQL/MX (page 3 of 3)

NonStop SQL/MP DATETIME Type	Supported in NonStop SQL/MX	Maps To ANSI Type
INTERVAL *start* to FRACTION(n)	Yes	INTERVAL *start* to SECOND(n)
All other INTERVAL start-end combinations	Yes	Same as in NonStop SQL/MP

*Limited support as dummy values in SELECT statements only

Interval Literals

As in NonStop SQL/MP, an SQL/MX interval literal is either year-month or day-time. If you are using SQL/MX DML statements with SQL/MP tables, NonStop SQL/MX recognizes fields created with the FRACTION keyword and maps them directly to the fractional precision of the SECOND field.

These are INTERVAL literals:

```
INTERVAL '7' DAY
INTERVAL '2-7' YEAR TO MONTH
INTERVAL '5:2:15:36.33' DAY TO SECOND(2)
```

An SQL/MP interval with a literal of DAY TO FRACTION(2) becomes in SQL/MX DAY TO SECOND(2).

Expressions and Functions

NonStop SQL/MP supports three types of expressions: value expressions, row-value constructors, and query expressions. NonStop SQL/MX supports SQL:1999 versions of value expressions, row-value constructors, and query expressions. For a complete list of SQL/MX expressions and functions, see the *SQL/MX Reference Manual*.

Table 3-2 lists only the SQL/MP functions that are obsolete or changed in NonStop SQL/MX:

Table 3-2. NonStop SQL/MP and NonStop SQL/MX Functions

NonStop SQL/MP Function	Implemented in NonStop SQL/MX?	Comments
COMPUTE_TIMESTAMP	No	
CURRENT	Changed	In NonStop SQL/MX, supported to return a range of date-time values only in NonStop SQL/MP stored text. Otherwise, in NonStop SQL/MX, same as CURRENT_TIMESTAMP. See mapping under Datetime Functions.
EXTEND	Changed	In NonStop SQL/MX, supported only for use in NonStop SQL/MP stored text. Otherwise, use the CAST specification in NonStop SQL/MX.
SETSCALE	Changed	In NonStop SQL/MX, you provide the scale when you declare your host variables in your program. See SETSCALE Function.

Except for the functions listed previously, NonStop SQL/MX offers all the functions of NonStop SQL/MP and adds many new functions.

Numeric Value Expressions and Functions

Both NonStop SQL/MX and NonStop SQL/MP offer an extension to SQL:1999, the exponentiation operator (**), but in results of computations of numeric expressions, NonStop SQL/MX might differ in precision and scale from the computations within NonStop SQL/MP.

Datetime Functions

In NonStop SQL/MP, the CURRENT function returns a current local DATETIME value, based on the specified start-end fields, such as CURRENT YEAR TO MONTH, CURRENT YEAR or CURRENT SECOND. CURRENT by itself is the equivalent of CURRENT YEAR TO FRACTION.

For the same results in NonStop SQL/MX, use the CURRENT_DATE, CURRENT_TIME, or CURRENT_TIMESTAMP function. In addition, CURRENT is an SQL/MX extension that maps to CURRENT_TIMESTAMP, equivalent to its result in NonStop SQL/MP. NonStop SQL/MX does not support the use of CURRENT with start-end fields. For this result, you should cast the appropriate form of SQL/MX CURRENT_DATE, CURRENT_TIME, or CURRENT_TIMESTAMP to the SQL/MP DATETIME type with the desired start-end fields. Table 3-3 compares SQL/MP and SQL/MX date-time functions:

Table 3-3. Mapping NonStop SQL/MP Datetime Functions To NonStop SQL/MX

NonStop SQL/MP	NonStop SQL/MX
CURRENT	CURRENT_TIMESTAMP
CURRENT YEAR TO DAY	CURRENT_DATE
CURRENT YEAR TO SECOND	CURRENT_TIMESTAMP(0)
CURRENT YEAR TO FRACTION(n)	CURRENT_TIMESTAMP(n)
CURRENT HOUR TO SECOND	CURRENT_TIME(0)
CURRENT HOUR TO FRACTION(n)	CURRENT_TIME(n)
CURRENT YEAR	CAST (CURRENT_DATE AS DATETIME YEAR)
CURRENT YEAR TO MONTH	CAST (CURRENT_DATE AS DATETIME YEAR TO MONTH)
CURRENT HOUR TO MINUTE	CAST (CURRENT_TIME AS DATETIME HOUR TO MINUTE)

Interval Value Expressions

NonStop SQL/MP allows an interval to be divided by an interval. For example, INTERVAL '6' DAY / INTERVAL '2' DAY results in an integer value of 3. NonStop SQL/MP does not support the CAST function for date-time and interval data types.

NonStop SQL/MX (and SQL:1999) supports the division (or multiplication) of an INTERVAL by a factor. For example, INTERVAL '6' DAY / 2 results in an interval of 3 days. If you want the result to be an integer and not an interval, convert or cast the result as an integer.

Mathematical Functions

NonStop SQL/MP does not support mathematical functions. NonStop SQL/MX supports many mathematical functions as extensions to the SQL-92 standard.

Sequence Functions

NonStop SQL/MP does not support sequence functions. In NonStop SQL/MX, sequence functions operate on ordered rows of a SELECT result table that includes a SEQUENCE BY clause.

Aggregate Functions

In a few cases, NonStop SQL/MP allows a subquery within an aggregate in a predicate. NonStop SQL/MX follows the ANSI standard and does not allow any subqueries in an aggregate.

In order to comply with ANSI standards, NonStop SQL/MX does not move aggregate predicates from the WHERE clause to a HAVING clause and does not move non-aggregate predicates from the HAVING clause to the WHERE clause, as NonStop SQL/MP does.

Predicates

Both NonStop SQL/MP and NonStop SQL/MX support the use of row-value constructors within predicates. However, the NonStop SQL/MP implementation is not SQL:1999 compliant. In NonStop SQL/MP, you are not required to enclose the sequence of expressions of a row-value constructor in parentheses. In NonStop SQL/MX, it is helpful (although not required) to do so. If you do not enclose the sequence of expressions in parentheses, your resulting queries might be ambiguous.

For example, in this predicate, consisting of a comparison predicate and a NULL predicate, SQL/MP row-value constructors use no parentheses:

```
WHERE a,b > 10,c AND p,q,r IS NULL
```

In contrast, in NonStop SQL/MX, the row-value constructors in this predicate use parentheses:

```
WHERE (a,b) > (10,c) AND (p,q,r) IS NULL
```

Use row-value constructors in comparison predicates, the IN predicate, the NULL predicate, and the quantified comparison predicates (ALL, ANY, SOME).

LIKE Predicate

Both NonStop SQL/MP and NonStop SQL/MX support the LIKE predicate. However, NonStop SQL/MP has an extension to the ANSI standard. It allows you to specify a TERMINATE character to indicate the end of the pattern within the pattern string. You can use this clause when the column value and the comparison value are of different lengths. NonStop SQL/MX supports the SUBSTRING and POSITION functions, which can be used for the same purpose as TERMINATE.

NonStop SQL/MX rewrites SQL/MP constraint and view text that uses the LIKE predicate and TERMINATE character:

Original SQL/MP predicate

```
column1 LIKE ?expr2 TERMINATE ?trm
column1 LIKE ?expr2 ESCAPE ?esc TERMINATE ?trm
```

Rewritten by NonStop SQL/MX

```
column1 LIKE SUBSTRING(?expr2 FROM 1
         FOR (POSITION (?trm in ?expr2) - 1))
column1 LIKE SUBSTRNG(?expr2 FROM 1
        FOR (POSITION(?trm IN ?expr2) - 1)) ESCAPE ?esc
```

Sort Operations

NonStop SQL/MP uses the FastSort sort/merge program for certain operations. These operations are described in the *SQL/MP Installation and Management Guide*. You use the EXPLAIN utility to determine if a sort occurs. The EXPLAIN reports any sort operations required to execute the query. You also use certain SORT-related DEFINEs to influence the effectiveness of some SQL operations.

NonStop SQL/MX does not use the FastSort program or the SORT-related DEFINEs. In addition, a scratch file in NonStop SQL/MX can overflow to another disk. The operation will not fail if a scratch file becomes full or if the disk becomes full; NonStop SQL/MX creates an additional scratch file either on the same or a different disk if the disk becomes full. The maximum number of scratch files that can be used is 2048. NonStop SQL/MX provides default settings that influence the choice of scratch disks.

SQL/MX parallelism is performed through ESPs only. Parallel sort can be forced through CONTROL QUERY SHAPE. When the SQL/MX optimizer runs parallel queries with ESPs, each ESP tries to use a different disk.

For additional information about how scratch disks are selected and influenced, see the Defaults section of the *SQL/MX Query Guide* and the *SQL/MX Reference Manual*.

SQL DML Statements

Most SQL/MP DML statements exist in NonStop SQL/MX. However, a few options and clauses have been dropped because of SQL:1999 ANSI changes. Table 3-4 lists the SQL/MP DML statements that differ in NonStop SQL/MX:

Table 3-4. SQL/MX and SQL/MP DML Statements

SQL/MP DML Statements	Implemented in NonStop SQL/MX?	Comments
CONTROL EXECUTOR	No	PARALLEL EXECUTION is ON by default in NonStop SQL/MX. See Default Settings on page 5-7.
CONTROL QUERY	Changed	See CONTROL QUERY Directive on page 3-23.
CONTROL TABLE	Changed	See CONTROL TABLE Directive on page 3-21; also see Forcing Query Execution Plans on page 5-8.

Table 3-4. SQL/MX and SQL/MP DML Statements

SQL/MP DML Statements	Implemented in NonStop SQL/MX?	Comments
INSERT	Changed	In NonStop SQL/MX, an ORDER BY clause can be used following the specification of the insert source. Also, multiple row-value constructors can be used following the VALUES keyword. See INSERT Statement on page 3-14.
SELECT	Changed	In NonStop SQL/MX, an SQL value expression can be used in a select list. Also, derived tables (or query expressions) can be used in a FROM clause. See SELECT Statement on page 3-15.
UPDATE	Changed	In NonStop SQL/MX, a scalar subquery can be used as the update value in a SET clause. See UPDATE Statement on page 3-16.

Nonaudited Tables

In both NonStop SQL/MP and NonStop SQL/MX, the HP NonStop Transaction Management Facility (TMF) product works only on audited tables, so a transaction does not protect operations on nonaudited tables. The simplest approach is to make all tables audited.

NonStop SQL/MP is designed to avoid any inconsistencies between the base table and an index for nonaudited tables. Because of its design, NonStop SQL/MX has a greater likelihood that such inconsistencies might occur.

Nonaudited tables are not protected by transactions and follow a different locking and error handling model than audited tables. In NonStop SQL/MX, certain situations such as DML error occurrences or utility operations with DML operations can lead to inconsistent data within a nonaudited table or between a nonaudited table and its indexes.

To avoid potential problems, do not create indexes on nonaudited tables. Do not run DDL or utility operations concurrently with DML operations on nonaudited tables. This table lists differences in how NonStop SQL/MP and NonStop SQL/MX handle nonaudited DML operations

Operation	NonStop SQL/MP	NonStop SQL/MX
Insert/update/delete with a user error during index maintenance	No inconsistencies	Potentially many rows affected
Insert/update/delete interrupted by internal program abend or CPU halt	At most one row	Potentially many rows affected

NonStop SQL/MX provides a default setting that can warn you at compile time of nonaudited index maintenance, called IUD_NONAUDITED_INDEX_MAINT. By default, operations that could lead to an inconsistency between base tables and indexes are

disabled. See the Defaults Table entry in the *SQL/MX Reference Manual* for further information.

INSERT Statement

RETURNING LASTSYSKEY Clause

NonStop SQL/MP supports the RETURNING LASTSYSKEY clause for the INSERT statement. NonStop SQL/MX does not support this option.

Insert Rows in Order

NonStop SQL/MX allows you to insert rows in order with an ORDER BY clause on the INSERT statement. When selecting rows from one table to be inserted into another, an ORDER BY clause can reorder the sequence of the insertion so that the inserts are done more efficiently. You cannot use ORDER BY in NonStop SQL/MX when the SELECT statement is a subquery. The ORDER BY clause cannot be used in embedded SQL programs. The ORDER BY clause is an SQL/MX extension to the ANSI-92 standard. For additional information about the ORDER BY clause, see the *SQL/MX Reference Manual*.

Inserting Multiple Rows

In NonStop SQL/MP, if you use the VALUES keyword, you can insert only one row. For example, this SQL/MP statement inserts a single row of values:

```
INSERT INTO JOB VALUES (1000, 'MANAGER');
```

In NonStop SQL/MX, if you use the VALUES keyword, you can insert multiple row-value constructors. For example, this SQL/MX statement inserts two rows into table JOB1:

```
INSERT INTO JOB1 VALUES (1000, 'MANAGER'), (2000, 'SUPERVISOR')
```

In general, in NonStop SQL/MX, each row to be inserted is specified as a list of value expressions or a row subquery. A value in a row can also be a scalar subquery. For example, this SQL/MX statement inserts two rows into table JOB:

```
INSERT INTO JOB VALUES (1000, 'MANAGER'), ((SELECT *
           FROM JOB1 WHERE JOBCODE = 2000));
```

In NonStop SQL/MP you usually perform multiple row inserts by setting a single row insert to run in a loop. Even though NonStop SQL/MP is inserting a single row, because it is looped, it will do this insert as a multiple row insert in DP2 using virtual sequential block buffering (VSBB).

NonStop SQL/MX’s architecture handles this differently and it is more efficient to use rowsets to perform multiple row inserts. If VSBB is on, EXPLAIN output will show INSERT_VSBB as the operator.

SELECT Statement

Table Reference Within FROM Clause

In NonStop SQL/MP, a table reference within a FROM clause of a SELECT statement can be a table, view, or joined table. The maximum number of tables that can be specified in a FROM clause is 16. This maximum includes the underlying base tables of views.

In NonStop SQL/MX, a table reference can also be a query expression or, stated more precisely, the derived table determined by the evaluation of a query expression. A query expression can be the UNION or JOIN of two (or more) table references. A query expression can also be a simple table—a table value constructor generated by a VALUES clause, TABLE clause, or a subquery.

In NonStop SQL/MX, an item in the select list can be a subquery:

```
SELECT (SELECT a FROM s WHERE b=t.b) FROM t;
```

In NonStop SQL/MX, a joined table can be used as a table reference in the FROM clause of a SELECT statement. An SQL:1999 joined table is specified by connecting the names of the two tables with the keywords NATURAL, CROSS, INNER, LEFT, or RIGHT preceding the JOIN keyword. NonStop SQL/MP supports LEFT and INNER joins only.

For example, the clause PARTS NATURAL JOIN ODETAIL (as a table reference in a FROM clause) joins rows only where the values of all columns that have the same name in both tables match. The clause PARTS NATURAL LEFT JOIN ODETAIL joins rows where the values of all columns that have the same name in both tables match plus rows from PARTS (the table to the left of the JOIN keyword) that do not meet this condition.

GROUP BY Clause and Numerical Arguments

In NonStop SQL/MP, the GROUP BY clause allows you to use numerical arguments, for example:

```
SELECT a + b FROM t GROUP BY 1;
```

NonStop SQL/MX does not allow the use of numerical arguments with the GROUP BY clause. However, you can use a derived query to achieve the same purpose as shown next:

```
SELECT c FROM (SELECT a + b FROM t) x(c) GROUP BY c;
```

Placement of GROUP BY Clause

NonStop SQL/MX requires the GROUP BY clause, if one exists, to precede a HAVING clause. In NonStop SQL/MP, you can place the clauses in any order.

UPDATE Statement

Source Value for a Column Referenced in the SET Clause

In NonStop SQL/MX, the source value for a column referenced in the SET clause of an UPDATE statement is an SQL value expression. In particular, a value expression can be a scalar subquery:

```
UPDATE JOB SET jobdesc=(SELECT jobdesc from JOB1
                        WHERE jobcode=2000)
            WHERE jobcode=2000;
```

In NonStop SQL/MP, the expression cannot be a subquery.

Parallel Plans

In NonStop SQL/MP, you cannot have a parallel plan where the UPDATE statement contains the WHERE CURRENT OF clause. In NonStop SQL/MX, you can have parallel plans with the WHERE CURRENT OF clause.

Access Options and Isolation Levels

NonStop SQL/MP allows you to specify access options for each DML statement. In NonStop SQL/MP, the *statement level* access options are BROWSE, STABLE, and REPEATABLE.

NonStop SQL/MX allows you to specify the isolation level at both the *statement* and *transaction* level. (*Isolation level*, SQL:1999 terminology, means the same as *access option*.) In NonStop SQL/MX, the isolation levels for both statements and transactions are READ UNCOMMITTED, READ COMMITTED, SERIALIZABLE (or REPEATABLE READ) and STABLE. Isolation levels set at the statement level take precedence over the transaction level settings. Isolation levels for statements are extensions to the ANSI standard. Isolation levels for transactions are SQL:1999 compliant; you set transaction isolation levels with the SET TRANSACTION statement.

Note. Unless you are writing applications for portability, you should use isolation levels at the statement level. The SET TRANSACTION statement might cause a dynamic recompilation of DML statements within the next transaction. Further, NonStop SQL/MX does not have the opportunity to optimize query execution plans as it does during static compilation. You can achieve better performance if you specify isolation levels for individual DML statements.

Isolation Levels Contrasted Between NonStop SQL/MP and NonStop SQL/MX

- BROWSE (NonStop SQL/MP) and READ UNCOMMITTED (NonStop SQL/MX) have the same implementation. This option is available only for the SELECT statement.

- STABLE (NonStop SQL/MP) and STABLE (NonStop SQL/MX) have the same implementation. This option is available only for the SELECT statement.
- REPEATABLE (NonStop SQL/MP and NonStop SQL/MX) and SERIALIZABLE (NonStop SQL/MX) have the same implementation; this option holds locks on data until the end of the transaction.

Note. If your SQL/MP application uses the BROWSE, STABLE, and REPEATABLE keywords, NonStop SQL/MX accepts the keywords as synonyms for READ UNCOMMITTED, STABLE, and SERIALIZABLE.

STABLE Access

How STABLE Access Works

In both SQL/MX and SQL/MP cursor operations using STABLE access, you are guaranteed another transaction cannot update the row and that the row can be safely updated without having to check versions of the row data (using UPDATE WHERE CURRENT OF or DELETE WHERE CURRENT OF). The number of rows that are locked depends on the value you have set for SYSTEM_DEFAULT attribute MAX_ROWS_LOCKED_FOR_STABLE_ACCESS.

If you specify STABLE access for an updatable SELECT statement, NonStop SQL/MX uses that access method. For other statements, NonStop SQL/MX changes the access method to READ COMMITTED. NonStop SQL/MX does not choose STABLE access for rowsets.

How READ COMMITTED Differs From STABLE Access

If you choose READ COMMITTED, read a row, and decide to update that row, in the time between reading the row and making the update, the value might change because no lock is acquired on the row. If your applications use STABLE access, you might want to evaluate that use to decide whether you should change the access to SERIALIZABLE or whether you should adjust your error handling procedures. You can rely on the fact that when you read a row, no other transaction can change the row while you are processing it. Thus, you are guaranteed not to cause any integrity issues when you do update the row.

READ COMMITTED also guarantees integrity but in a different way. If you update the row acquired using READ COMMITTED access and another transaction has updated it since you first read it, a warning is returned to the application on the update. You can check the warning code returned and take appropriate action (abort the transaction if necessary, display an appropriate message, retry the read and update, and so on). By changing your error handling procedures, you can benefit from the increased concurrency.

STABLE Access and EXCLUSIVE MODE

In both NonStop SQL/MX and NonStop SQL/MP, if you use STABLE access with EXCLUSIVE MODE, you do not get an error, because the next transaction must wait for access to the row.

Transaction Management

A TMF transaction is the basic recoverable unit in case of a failure or transaction interruption. TMF transactions can be defined during an MXCI session or in a host program. The typical order of events is:

1. Transaction is started.
2. Database changes are made.
3. Transaction is committed.

If the changes cannot be made or you do not want to complete the transaction, your program can abort the transaction so the database is rolled back to its original state.

Starting and Ending a Transaction

In NonStop SQL/MX and NonStop SQL/MP, to ensure that a sequence of statements either executes successfully or not at all, define one transaction consisting of these statements by using the BEGIN WORK and COMMIT WORK statements. You can abort a transaction with the ROLLBACK WORK statement. BEGIN WORK is an extension to the ANSI standard.

User-Defined or Explicit Transactions

Transactions you do not start through SQL, such as TMF, MXCS, or inherited transactions, are called user-defined, or explicit, transactions. You must commit or roll back these transactions.

If an error occurs in an explicit transaction for an INSERT, UPDATE, or DELETE statement on audited tables, the transaction is not aborted automatically. However, your data can be in an inconsistent state within a query.

System-Defined or Implicit Transactions

NonStop SQL/MP has implicit transactions in SQLCI but not in embedded statements.

NonStop SQL/MX also has implicit transactions. It initiates a transaction when it encounters an SQL statement if a BEGIN WORK statement has not been specified and if the statement references an audited table. Most DML statements initiate transactions implicitly at the start of execution. NonStop SQL/MX does not initiate a transaction for a SELECT statement with READ UNCOMMITTED access.

Even if a transaction is initiated implicitly, you must end a transaction explicitly with the COMMIT WORK statement or the ROLLBACK WORK statement so that either the entire SQL statement executes or none of it is does.

Auto-Abort

NonStop SQL/MX Release 1.8 automatically aborted transactions if an error occurred while performing a SQL DELETE, INSERT, UPDATE, or DDL statement. By default, NonStop SQL/MX Release 2.0 does not automatically abort transactions following an error. The option to abort the transaction following an error must be determined by the application program.

You can set a CONTROL QUERY DEFAULT, UPD_ABORT_ON_ERROR, to make SQL/MX Release 2.0 behave like SQL/MX Release 1.8 and automatically abort transactions following an error. See the description of this control query default in the *SQL/MX Reference Manual* for details.

SQL/MP Behavior

In NonStop SQL/MP, you can use row-not-found warnings or row-already-exists errors to determine whether to insert or update single rows. You can do an INSERT to find out if the row already exists, ignore the error, and follow it with an UPDATE statement to update the row. Or, do an UPDATE on a row to see if it exists, and when no row is found, use an INSERT statement to insert the row. This process is sometimes called an “UPSERT.”

SQL/MX Release 2.0 enables you to use row-not-found and row-already exists errors the same way, by leaving the CONTROL QUERY DEFAULT UPD_ABORT_ON_ERROR set to OFF, the default. See the description of this control query default in the *SQL/MX Reference Manual* for details.

Inherited Transactions

In NonStop SQL/MP and NonStop SQL/MX, if a transaction is started at some point during the execution of your program (for example, in TMF or MXCS), you do not need to explicitly start a new SQL transaction. SQL detects that a transaction exists and inherits that transaction. If an error occurs in an inherited transaction, NonStop SQL/MX changes the transaction state to doomed (SQLSTATE “25000”). A doomed transaction is not aborted. However, to proceed, you must manually abort the transaction. NonStop SQL/MX provides error messages until then.

SET TRANSACTION Statement

In NonStop SQL/MX, to automatically end a transaction after a statement successfully executes, use the AUTOCOMMIT option within a SET TRANSACTION statement in both MXCI and embedded SQL. This option specifies whether or not NonStop SQL/MX commits at the end of statement execution and applies to any statement for which SQL initiates a transaction (implicit transaction). This option does not apply to transactions you start with the BEGIN WORK statement or transactions started by services, such as

TMF or MXCS, or inherited from other processes. NonStop SQL/MP provides a similar SQLCI command, called AUTOWORK.

See Cursor Operations (Embedded SQL) on page 3-21 for information about using the AUTOCOMMIT option with cursors.

The next statement sets the AUTOCOMMIT option to ON:

```
SET TRANSACTION AUTOCOMMIT ON;
```

If this option is set to ON, at the end of each statement execution, NonStop SQL/MX automatically commits changes made to the database. AUTOCOMMIT is set to ON by default at the start of an MXCI session.

If this option is set to OFF, the current transaction remains active until the end of the MXCI session or, for embedded SQL, the end of program execution, at which time the transaction is aborted—unless you explicitly commit or roll back the transaction. The default is OFF for embedded SQL.

You can use the MXCI command ENV to check the state of the transaction.

AUTOCOMMIT Example

In the next example, the SET TRANSACTION statement sets AUTOCOMMIT to ON. The first INSERT statement starts an implicit transaction. When the INSERT finishes executing, the transaction, consisting of one inserted row, is committed. The next INSERT statement occurs, and after it finishes, its transaction is committed. You start a user-defined (explicit) transaction with the BEGIN WORK statement. A series of UPDATE statements are contained in this user-defined transaction, and after the COMMIT WORK statement is issued, the updates are committed. Remember that the AUTOCOMMIT option is still set to ON, so that when the next DML statement is initiated, an implicit transaction is started, and when the statement successfully finishes, the transaction is committed.

```
SET TRANSACTION AUTOCOMMIT ON;

INSERT ...;        --Implicit transaction started
                   --Row committed
INSERT ...;        --Implicit transaction started
                   --Row committed
                   --AUTOCOMMIT STILL ON
```

```
BEGIN WORK;        --Explicit transaction started

UPDATE ...
UPDATE ...         --AUTOCOMMIT OFF
COMMIT WORK;
                   --Explicit transaction committed
                   --AUTOCOMMIT STILL ON


INSERT ...;        --Implicit transaction started
                   --Row committed
INSERT ...;        --Implicit transaction started
 .                 --Row committed
 .
 .
```

Cursor Operations (Embedded SQL)

If you are performing cursor UPDATE or DELETE operations and have the AUTOCOMMIT option turned on, the transaction is committed and the cursor closed at the end of the first UPDATE WHERE CURRENT OF or DELETE WHERE CURRENT OF. You cannot perform another fetch without reopening the cursor. You should turn off the AUTOCOMMIT feature for cursor operations. When you complete your cursor operations, you can turn the AUTOCOMMIT feature on again.

CONTROL TABLE Directive

In NonStop SQL/MP, you can use the CONTROL TABLE directive for many purposes but mainly to specify performance-related options for DML accesses to a table or view. In NonStop SQL/MP, the CONTROL TABLE directive affects decisions the SQL compiler makes about how to execute DML statements. In NonStop SQL/MX, because the compiler and executor have been redesigned, many directives are no longer necessary.

In NonStop SQL/MP, you can enter multiple options on a single CONTROL TABLE statement. In NonStop SQL/MX, you can enter only one option on a single CONTROL TABLE statement.

Table 3-5 lists which SQL/MP controls are not implemented in NonStop SQL/MX or are implemented differently:

Table 3-5. CONTROL TABLE Directives

SQL/MP CONTROL TABLE Directives	Implemented in NonStop SQL/MX?	Comments
ACCESS PATH	Changed	Access paths can be forced using CONTROL QUERY SHAPE. See Forcing Query Execution Plans on page 5-8.
JOIN METHOD	Changed	Join methods can be forced using CONTROL QUERY SHAPE. See Forcing Query Execution Plans on page 5-8.
JOIN SEQUENCE	Changed	Join sequence can be defined with a CONTROL QUERY DEFAULT setting, JOIN_ORDER_BY_USER. See CONTROL QUERY Directive on page 3-23.
TIMEOUT	Changed	The SQL/MP CONTROL TABLE * TIMEOUT statement combines the functions of the CONTROL TABLE TIMEOUT and SET TABLE * TIMEOUT statements.
OPEN PARTITIONS	No	
SEQUENTIAL	No	
SEQUENTIAL BLOCKSPLIT	No	
SKIP UNAVAILABLE PARTITION	No	
SYNCDEPTH	No	

Use the MXCI command, SHOWCONTROL, to see which controls are in effect.

In NonStop SQL/MX, the CONTROL TABLE directive requires that you enclose all arguments in single quotes. In NonStop SQL/MP, the CONTROL TABLE directive has no such restriction.

In NonStop SQL/MP, the control TIMEOUT allows you to use a negative value (-1), which is essentially wait forever. This value is not allowed in NonStop SQL/MX.

The NonStop SQL/MP CONTROL TABLE directive enables you to adjust the timeout value for compile time. NonStop SQL/MX provides compile time directives for CONTROL TABLE TIMEOUT and CONTROL TABLE STREAM_TIMEOUT. NonStop SQL/MX provides for run-time timeout with the SET TABLE TIMEOUT statement as described in the *SQL/MX Reference Manual* and *SQL/MX Programming Guide for C and COBOL*.

See the *SQL/MX Reference Manual* for additional information about the CONTROL TABLE directive.

CONTROL QUERY Directive

In NonStop SQL/MP, the CONTROL QUERY compiler directive controls plans for queries. Options specify whether to resolve names at execution time or startup time, whether to include hash join algorithms considered for executing queries, and whether to optimize query response time for returning a few rows or all rows. In NonStop SQL/MX, the CONTROL QUERY directive has been changed to CONTROL QUERY DEFAULT. Many options have been changed.

The next table lists the SQL/MP CONTROL QUERY directives. More information about default settings is provided in the *SQL/MX Reference Manual* and the *SQL/MX Query Guide*.

SQL/MP CONTROL QUERY Directive	How NonStop SQL/MX Affects the Same Functionality	Comments
BIND NAMES	Internally handled in the compiler. No external control.	
HASH JOIN	CONTROL QUERY SHAPE to force	See Forcing Query Execution Plans on page 5-8.
INTERACTIVE ACCESS ON	Not implemented.	
MDAM	CONTROL TABLE or CONTROL QUERY SHAPE to force off	ON by default; see Forcing Query Execution Plans on page 5-8.

Query Caching

In NonStop SQL/MP, you control statement and query caching with MXCS and SQL/MP configuration attributes, system defaults, and control statements. These settings persist until a user changes a Guardian user ID, a server is reinitialized, or a rollback condition occurs. Caching takes place at a high level.

In NonStop SQL/MX, caching is controlled by control query default statements. Caching takes place within each SQL/MX MXCMP compiler session and is applicable to query compilation requests in that session. MXCS, JDBC, and MXCI clients have their own MXCMP sessions. Each MXCMP session has caching turned on by default. MXCS and JDBC/MX clients use control statements that you can store in a named collection linked to a user and stored in configuration tables for that user, so they are always set for that user.

See the *SQL/MX Reference Manual* for details about these settings and the *SQL/MX Query Guide* for more information about queries.

This table compares caching features in NonStop SQL/MP and NonStop SQL/MX:

Feature	NonStop SQL/MP	NonStop SQL/MX
Cache location	In ODBC/MP server	In MXCMP compiler
Cache setting	Disabled by default	Enabled by default
Cache size limit	32,767 statements	4,194,303 KB of memory
Efficiency increase from cache hit	High	Medium
Effect of schema change on cached query	Flushes cache.	Flush and re-cache transparently.
Caching considerations	Caching is discouraged for SELECT queries with ORDER BY clauses. An optional feature of statement cache supports caching parallel plans and allocated ESPs. To avoid resource exhaustion, use caching only where ESP sharing is enabled and the number of statements cached are small.	Complex queries and DSS-type queries are currently not cached.
Location of caching control	ODBC/MP user's profile.	SQL/MX’s MXCS session.

Feature	NonStop SQL/MP	NonStop SQL/MX
When caching change takes effect	On next system configuration refresh.	At next query compile.
Ownership of object	Cache per ODBC/MP server.	Cache per MXCMP process.
When cache is flushed	When Guardian user ID changes. CONTROL statements only affect the next statement compilation, which is avoided when a statement is used from cache. Cache is not preserved over critical condition termination. All statements are flushed at server reinitialization.	On demand. User requests cache size be reduced to zero by issuing `CONTROL QUERY DEFAULT QUERY_CACHE '0'` statement.
Cache statistics	Logged in MP table.	In-memory counters.
When statistics are saved and reported	Performed by server, either at statement close or at session disconnect.	In-memory counters are retrieved when user issues `SELECT … FROM table(querycache()) WHERE …` and `SELECT … FROM table(querycacheentries()) WHERE …` statements.

Feature	NonStop SQL/MP	NonStop SQL/MX
Cache counters	Number of statements, lookups, hits, drops. Extended to indicate whether statement was not cached because it is parallel, and parallel was not activated.	In-memory counters are: Number of entries, pinned entries, compiles, cachable compiles by phase, recompiles, retries, cachables, hits by phase, pinned hits by phase, displaced, cachable-but-too-large, current size, max size, average plan size, max victims, optimization level, pinning state. Per entry: hits, size, text, plan ID, row ID, phase, optimization level, catalog name, schema name, pin, number of parameters, parameter types, plan length, compilation time, average hit time, required shape, isolation level, access mode, autocommit, rollback mode setting.
Types of queries that are cachable	ExecDirect or Prepare user SQL DML statements, without dynamic parameters.	ExecDirect or Prepare user SQL DML statements.
When constant values are bound to a query's system-introduced parameters	Just before execute step in ODBC/MP server. Literals are transformed into parameters during the statement translate stage.	After generate step in MXCMP compiler.

Query Type Comparison

Simple data manipulation language (insert, update, delete, select) statements without predicates, or with key equality predicates that are guaranteed to affect at most one row, are cached in both NonStop SQL/MP and NonStop SQL/MX. An equality predicate is cachable if one of the operands is a literal and the other operand is a fully specified clustering (or primary) key.

Data manipulation statements with inequality predicates, nonkey predicates, subqueries, multirow and multiway joins are cached in NonStop SQL/MP but not in NonStop SQL/MX. Many decision support type queries are cached in NonStop SQL/MP but not in NonStop SQL/MX.

Table 3-6 describes various statements, with examples, and indicates whether they are cached by NonStop SQL/MP and NonStop SQL/MX:

Table 3-6. Query Type Comparison (page 1 of 3)

Type of Query	NonStop SQL/MP	NonStop SQL/MX
Tuple inserts: `INSERT INTO t1 VALUES(1,1,1)`	Y	Y

Insert select with key equality predicate:
`INSERT INTO t1 SELECT * FROM s`
`  WHERE key_col_a=1`
	Y	Y

Insert select with column comparison predicate:
`INSERT INTO t1 SELECT * FROM t2`
`  WHERE key_col_a=key_col_b`
	Y	N

Insert select with any nonkey predicate:
`INSERT INTO t1 SELECT * from t2`
`  WHERE nonkey_col<1`
	Y	N

Insert select with subquery:
`INSERT INTO t1 SELECT * FROM t2`
`  WHERE col_a IN(SELECT col_b FROM u)`
	Y	N

Updates with key equality predicates: `UPDATE t1 SET col.a=1 WHERE key_col=2`	Y	Y

Updates with column comparison predicate:
`UPDATE t1 SET col.a=1`
`  WHERE key_col_a=key_col_b`
	Y	N

Updates with any nonkey predicate: `UPDATE t1 SET col.a=1 WHERE nonkey_col_a>2`	Y	N

Updates with subquery:
`UPDATE t1 SET a=1 WHERE col_a IN`
`  (SELECT col_b FROM t2 WHERE col_c=1)`
	Y	N

Table 3-6. Query Type Comparison (page 2 of 3)

Type of Query	NonStop SQL/MP	NonStop SQL/MX
Deletes with key equality predicates: `DELETE FROM t1 WHERE KEY=1`	Y	Y
Deletes with column comparison predicate: `DELETE FROM t1 WHERE key_col_a=key_col_b`	Y	N
Deletes with any nonkey predicate: `DELETE FROM t1 WHERE nonkey_col_a <> 1`	Y	N

Deletes with subquery:
`DELETE FROM t1 WHERE col_a IN`
`(SELECT col_b FROM t2 WHERE col_c=1)`
	Y	N

Selects with key equality predicate: `SELECT * FROM t1 WHERE key_col_a=1`	Y	Y
Select with column comparison predicate: `SELECT * FROM t1 WHERE key_col_a=key_col_b`	Y	N
Selects with a nonkey predicate: `SELECT * FROM t1 WHERE nonkey_col<1`	Y	N

Selects with a subquery:
`SELECT * FROM t1`
`WHERE col_a IN (SELECT col_b FROM t2)`
	Y	N

Single row two-way joins:
`SELECT * FROM t1,t2`
`WHERE t1.key_col_a=t2.key_col_b`
`AND t1.key_col_a=1`
`AND t2.key_col_b=2`
	Y	Y

Selects with multirow two-way joins:
`SELECT * FROM t1,t2`
`WHERE t1.key_col_a=t2.key_col_b`
	Y	N

Selects with multirow, multiway nonequi joins:
`SELECT * FROM t1,t2,t3`
`WHERE t1.key_col_a>t2.key_col_b`
`AND t2.key_col_b<t3.key_col_c`
	Y	N

Aggregates with key equality predicate: `SELECT COUNT(*) FROM t1 WHERE key_col_a=1`	Y	Y

Aggregates with column comparison predicate:
`SELECT COUNT(*) FROM t1`
`WHERE key_col_a=key_col_b`
	Y	N

Aggregates with nonkey predicate: `SELECT COUNT(*) FROM t1 WHERE nonkey_col<1`	Y	N

Table 3-6. Query Type Comparison (page 3 of 3)

Type of Query	NonStop SQL/MP	NonStop SQL/MX

Aggregates with subquery:
`SELECT COUNT(*) FROM t1`
`  WHERE col_a IN (SELECT col_b FROM t2)`
	Y	N

Most DSS queries:
`SELECT COUNT(*) FROM t1`
`  WHERE col_c<=1 GROUP BY col_a`
	Y	N

Statements that reference date-time data types	Y	N
Data Definition Language statements	N	N
Data Control Language statements	N	N

Versioning

NonStop SQL/MX database objects inherit their schema's version, regardless of what features they use. SQL/MP database objects have a feature-based object version.

4 Embedded SQL

As in NonStop SQL/MP, to access an SQL/MX database, you can execute SQL statements interactively by using MXCI (the NonStop SQL/MX equivalent to SQLCI) or programmatically by embedding, or including, SQL statements in a host program. In NonStop SQL/MX, you can write the host programs by using embedded SQL in the ANSI C, C++, COBOL, or Java programming languages.

This section discusses differences in the use of embedded SQL in C, C++, and COBOL programs between NonStop SQL/MP and NonStop SQL/MX. The main differences are in the diagnostics areas for error handling and in the descriptor areas for dynamic SQL programming. For detailed information about the use of embedded SQL in C, C++, and COBOL programs in NonStop SQL/MX, see the *SQL/MX Programming Guide for C and COBOL*. For information about embedding SQL/MX statements in Java programs, see the *SQL/MX Programming Manual for Java*.

This section is organized into these categories:

Host Variable Declarations

NonStop SQL/MP allows the declaration of character host variables by using arithmetic operations to determine the size of the host variable array. NonStop SQL/MX does not allow arithmetic operations to determine the size of the array. In NonStop SQL/MX, host variables must have constants specifying the size of the array.

Many SQL/MP programs declare host variables from structures that are defined outside of the DECLARE SECTION. NonStop SQL/MX requires that structure definitions for host variables must be declared between BEGIN DECLARE SECTION and END DECLARE SECTION.

SETSCALE Function

In NonStop SQL/MP, using the C programming language, use the SETSCALE function to set the scale of certain data types. You must use the SETSCALE function whenever the host variable is used in the query. In NonStop SQL/MX, scale information for a host variable is supplied when the host variable is declared inside the BEGIN DECLARE SECTION.

SQL/MP Example

This C program fragment uses SETSCALE with an INSERT statement to create a new row with the value 98.34 in the PARTS.PRICE column after storing the value in host variable `unit_price`. The value is multiplied by 100 for storing as a whole number.

```
unit_price = 9834;

EXEC SQL INSERT INTO =PARTS (PRICE)
   VALUES (SETSCALE (:unit_price, 2));
```

NonStop SQL/MX Example

You can declare a host variable as NUMERIC (p,s), where p is the precision and s is the scale. You do not use the SETSCALE function when you use the host variable. For example, you can declare a numeric(4,2) host variable as:

```
NUMERIC(4,2) unit_price;
```

The same scale applies when a value is fetched. For this example, the fetched value has an implicit scale of 2.

Variable-Length Character Data in Embedded SQL C Programs

In NonStop SQL/MP, the VARCHAR data type defined for a column represents one data element. However, suppose the column CUSTNAME is defined as VARCHAR(26) in NonStop SQL/MP. The SQL/MP INVOKE directive generates a structure of the form:

```
struct
{
  short len;
  char val[27];
} custname;          /* SQL/MP variable-length character data */
```

where `len` is a numeric data item that represents the length, and `val` is a fixed-length character data item for the string plus an extra byte for the null terminator. In NonStop SQL/MP, you can also explicitly declare a structure of the preceding form as a host variable for a VARCHAR column.

In NonStop SQL/MX, in addition to the `char` data type, you can declare the VARCHAR data type for host variables within embedded SQL C programs (SQL:1999 provides the VARCHAR data type for C programs). For example, this declaration is for a column up to 26 bytes:

```
VARCHAR custname[27];/* SQL/MX variable-length character data */
```

As in NonStop SQL/MP, the last byte of the array is for the null terminator. You must declare the array one byte larger to allow for the null terminator. The null terminator follows the data inserted, so if you insert a six-character string into a VARCHAR host variable specified as 27 characters, the null terminator is the seventh character.

INVOKE Directive

In NonStop SQL/MP, use the INVOKE directive interactively through SQLCI to create host variable declarations in an EDIT file. The EDIT file can then be included in an embedded SQL program.

In NonStop SQL/MX, use the INVOKE directive to create host variable declarations only directly in an embedded SQL program. Use the INVOKE command in MXCI to display the table and column names and data types associated with the columns; you cannot invoke the format of another language (such as COBOL).

For either kind of table, INVOKE does not support the DATEFORMAT clause. INVOKE returns the ANSI DATETIME data type. The SQL DATEFORMAT function returns a character string. They are not compatible and cannot be used interchageably.

Data Type Conversion

For information about the valid conversions you can perform with CAST, see the *SQL/MX Reference Manual*.

Host Variables With Date-time Data Types

In NonStop SQL/MX, you can convert data to the data type you specify by using the CAST specification for any expression, whether or not you are using a parameter. NonStop SQL/MX replaces the SQL/MP TYPE AS clause with the CAST specification. In previous releases of NonStop SQL/MX, you were required to cast DATE, TIME, TIMESTAMP, and INTERVAL data types. In NonStop SQL/MX Release 2.0 you can directly declare them as host variables.

In NonStop SQL/MP, use the CAST function to associate a character or numeric data type with a dynamic input parameter. Use the TYPE AS clause in the parameter specification to associate a date-time or interval data type with a parameter.

Indicator Variables

As in NonStop SQL/MP, the primary use of the indicator variable in NonStop SQL/MX is to indicate whether the value supplied is null. If the source value is null, the value of the indicator variable is set to less than zero.

However, in NonStop SQL/MX, the indicator variable has another use. If an indicator variable is associated with a target character host variable in the INTO clause of a SELECT or FETCH statement, and the length of the target variable is smaller than the value returned, the truncated string is placed into the host variable, the indicator variable is set to the length of the source string, and a warning message is issued. Otherwise, if the length of the target variable is sufficient, the indicator variable is set to 0.

For example, suppose the columns PARTDESC and QTY_AVAILABLE in the PARTS table allow null. After the SELECT statement executes, this C example tests the indicator variable for null or a truncated value:

```
EXEC SQL SELECT partnum, partdesc, price, qty_available INTO
         :parts_rec.partnum,
         :parts_rec.partdesc
            INDICATOR :parts_rec.partdesc_i,
         :parts_rec.price,
         :parts_rec.qty_available
            INDICATOR :parts_rec.qty_available_i,
    FROM sales.parts
    WHERE partnum = 300380 AND partnum = 2402 ;
...
if (parts.qty_available_i < 0) handle_null_value();
if (parts.partdesc_i > 0 ) handle_truncated_value();
...
```

In NonStop SQL/MP, the indicator variable is not set to the source length in the case of truncation. It is set to 0 if it is a nonnull value and if the value is null, the indicator variable is set to -1.

Embedded SQL Statements

Most SQL/MP embedded SQL statements exist in NonStop SQL/MX. However, a few of the statements, options, and clauses have been dropped because of SQL:1999 ANSI changes.

Table 4-1. Embedded SQL Statements

SQL/MP Embedded SQL Statements and Directives	Implemented in NonStop SQL/MX?	Comments
CLOSE	Yes	
DECLARE CURSOR	Changed	See Updatable Cursors.
DESCRIBE	Changed	NonStop SQL/MX uses an SQL descriptor area, as defined by ANSI SQL:1999. See Descriptor Area.
DESCRIBE INPUT	Changed	NonStop SQL/MX uses an SQL descriptor area, as defined by ANSI SQL:1999. See Descriptor Area.
EXECUTE	Changed	No RETURNING clause in NonStop SQL/MX.
EXECUTE IMMEDIATE	Yes	
FETCH	Changed	NonStop SQL/MX requires the number of host variables in the fetch list to be the same as the number of values returned in the cursor specification.

Table 4-1. Embedded SQL Statements

SQL/MP Embedded SQL Statements and Directives	Implemented in NonStop SQL/MX?	Comments
FREE RESOURCES	No	
INCLUDE STRUCTURES	No	See Descriptor Area, Diagnostics Area, Statistics Area.
INCLUDE SQLCA	No	See Diagnostics Area.
INCLUDE SQLDA	No	See Descriptor Area.
INCLUDE SQLSA	No	See Statistics Area.
INSERT	Changed	No RETURNING clause in NonStop SQL/MX. See INSERT Statement on page 3-14.
INVOKE	Changed	No FORMAT clause in NonStop SQL/MX. See INVOKE Directive.
OPEN	Changed	NonStop SQL/MX uses an SQL descriptor area, as defined by ANSI SQL:1999. See Descriptor Area.
Positioned DELETE	Yes	
Positioned UPDATE	Changed	NonStop SQL/MX allows this statement to be dynamic.
PREPARE	Yes	SQL/MX FROM clause does not allow quoted strings.
RELEASE	No	
SELECT INTO	Changed	See SELECT Statement.
WHENEVER	Changed	Usage changes. See WHENEVER Declarative.
Positioned DELETE	Yes	

EXECUTE Statement

In NonStop SQL/MX, input data is supplied by the USING clause that specifies either a list of host variables or an SQL descriptor area. Output data is provided by the INTO clause that specifies either a list of host variables or an SQL descriptor area. See Descriptor Area.

NonStop SQL/MP implements the RETURNING clause for output data. NonStop SQL/MX does not provide a RETURNING clause.

In both NonStop SQL/MX and NonStop SQL/MP, if you perform a PREPARE and an EXECUTE of an SQL UPDATE, INSERT or DELETE statement on a non-audited table, NonStop SQL/MX does not close the table and will not release record and table locks. You can use FUP LISTLOCKS to check if locks are kept on that table by your MXCI session after completion of the EXECUTE statement, then manually execute an

UNLOCK command or exit the MXCI session. You can also use standalone UPDATE, INSERT or DELETE statements instead of using PREPARE and EXECUTE. Only SQL/MP tables can be non-audited.

Updatable Cursors

In NonStop SQL/MP, delete operations using a cursor do not require the FOR UPDATE OF clause of the DECLARE CURSOR statement; you must use this clause when you update rows, but it is optional when you delete rows. In NonStop SQL/MX, both update and delete operations require an updatable cursor.

In NonStop SQL/MP, the cursor defaults to read only unless specified FOR UPDATE. In NonStop SQL/MX, within the DECLARE CURSOR statement, the optional FOR clause has this form:

```
FOR {READ ONLY | UPDATE [OF colname [,colname]...]}
```

By using the FOR clause, you can specify whether the cursor is FOR READ ONLY (read-only cursor) or FOR UPDATE OF (updatable cursor). If you do not specify the FOR clause, and if ORDER BY or GROUP BY is specified for the cursor or if the query expression defining the cursor specifies a read-only table, the cursor defaults to read-only. It is the query that determines that the table is read-only, not the table's attributes. Otherwise, the cursor defaults to updatable (without a column list). If you know that you are not going to update a cursor, specify FOR READ ONLY.

FETCH Statement

In NonStop SQL/MP, when a FETCH statement executes, if the number of host variables is different from the number of columns in the result table of the declared cursor, a warning is issued, and NonStop SQL/MP returns the number of values in the shorter list.

In NonStop SQL/MX, the number of host variables must be the same as the number of columns in the result table or an error is returned.

Positioned UPDATE Statement

In NonStop SQL/MP, you can execute the UPDATE ... WHERE CURRENT OF statement for static SQL only.

In NonStop SQL/MX, you can execute the UPDATE ... WHERE CURRENT OF statement by static SQL and dynamic SQL.

Dynamic SQL

Dynamic SQL allows a program to construct, compile, and execute all or part of an SQL statement at run time. NonStop SQL/MP provides a descriptor area, referred to as the SQLDA, containing information about input parameters and output variables in dynamic SQL statements.

NonStop SQL/MX provides for the allocation of a descriptor area with the use of SQL:1999 dynamic SQL statements. This descriptor area consists of multiple item descriptor areas, together with a count of the number of those item descriptor areas. For further information about using the descriptor area, see Descriptor Area on page 4-7.

The next list summarizes the dynamic SQL statements that you use with an SQL/MX descriptor area. For information about the syntax and use of these statements, see the *SQL/MX Reference Manual*.

ALLOCATE DESCRIPTOR	Allocates an input or output SQL descriptor area.
DEALLOCATE DESCRIPTOR	Deallocates an SQL descriptor area.
GET DESCRIPTOR	Retrieves information from an SQL descriptor area—the COUNT of the item descriptor areas and specified fields in the areas.
SET DESCRIPTOR	Modifies information in an SQL descriptor area.
PREPARE	Prepares (compiles) a dynamic SQL statement for subsequent execution by an EXECUTE statement.
DEALLOCATE PREPARE	Deallocates a prepared statement and returns the system resources used by the statement; also allows you to reuse the name of the statement.
DESCRIBE [OUTPUT]	Stores in the output descriptor area information about output values (usually SELECT columns) from a prepared statement.
DESCRIBE INPUT	Stores in the input descriptor area information about dynamic input parameters for a prepared statement.
EXECUTE	Executes a prepared dynamic SQL statement.
EXECUTE IMMEDIATE	Prepares (compiles) and executes a dynamic SQL statement.

Descriptor Area

NonStop SQL/MP provides a descriptor area (SQLDA) that enables a dynamic SQL application to exchange information with the database about input parameters and output columns. You can declare the descriptor area by using the INCLUDE SQLDA directive.

NonStop SQL/MX does not provide the SQLDA as does NonStop SQL/MP and does not provide the INCLUDE SQLDA directive. However, NonStop SQL/MX does provide an SQL descriptor area for dynamic SQL, which consists of multiple item descriptor areas, together with a count of the number of those item descriptor areas.

In NonStop SQL/MX, you must allocate and deallocate input and output SQL descriptor areas by using the ALLOCATE DESCRIPTOR and DEALLOCATE

DESCRIPTOR statements. After the parameters have been described (by using the DESCRIBE statement as in NonStop SQL/MP), you can modify and retrieve information from the descriptor areas only by using the SET DESCRIPTOR and GET DESCRIPTOR statements, respectively.

If you have dynamic input parameters in a prepared SQL statement, you must code the appropriate 3GL statements and use SET DESCRIPTOR to set the values in the descriptor area. If you have more than one dynamic input parameter, you can identify the values by their position in the prepared SQL statement. If you are using descriptor areas for input parameters, you must use the CAST specification to specify the appropriate data type.

In NonStop SQL/MX, you should give input parameters a defined data type provided in the descriptor area by using the CAST specification. The CAST specification is not needed for MXCI parameters or for parameters used in dynamic cursors in embedded SQL programs.

For examples that use dynamic SQL statements, see the *SQL/MX Reference Manual*.

Error Handling

The major difference in error handling between NonStop SQL/MP and NonStop SQL/MX relates to the SQL:1999 standard, as described next.

Format of Error Codes

NonStop SQL/MP provides an integer-valued SQLCODE as the mechanism to send error information from the database to the application.

NonStop SQL/MX supports both an SQLCODE mechanism and the new SQLSTATE codes, as specified in the SQL:1999 standard. However, the SQLCODE values generated by NonStop SQL/MX are different from those generated by NonStop SQL/MP. For portable applications, you should use SQLSTATE. Because of the overhead associated with SQLSTATE, you will most likely use it only for error handling in ANSI-compliant applications. For information about SQLSTATE and error handling in NonStop SQL/MX, see the *SQL/MX Programming Guide for C and COBOL*.

Equivalent SQLCODE and SQLSTATE Values

Error	NonStop SQL/MP SQLCODE	NonStop SQL/MX SQLCODE	NonStop SQL/MX SQLSTATE
SQL_NO_ERROR	0	0	00000
SQL_NO_ROWS_MATCH	100	100	02000
SQL_TOO_MANY_MEET_CRITERIA	-8222	-8401	21000
SQL_DUPLICATE_ENTRY	-8227	-8102	23000

SQL_PARTITION_NOT_AVAIL	-8239	*	*
SQL_FILE_SYSTEM_ERROR	-8300	-8551	-
SQL_NO_INDICATOR_VAR	-8423	-8420	22002
SQL_LOST_OPEN_WARN	8204	8574	-
SQL_LOST_OPEN_ERROR	-8204	-8574	-

* Not available. The current release of NonStop SQL/MX does not support the SQL/MP CONTROL TABLE SKIP UNAVAILABLE PARTITION option.

WHENEVER Declarative

In NonStop SQL/MP, the WHENEVER declarative specifies an action to take when an error, warning, or no-rows-found condition occurs. Further, NonStop SQL/MP supports all three actions: CONTINUE, GO TO (or GOTO), and PERFORM (in COBOL) or CALL (in C).

NonStop SQL/MX continues to support all three conditions, although SQL:1999 supports only SQLERROR and NOT FOUND.

In addition, NonStop SQL/MX continues to support all three actions, although SQL:1999 supports only CONTINUE and GO TO. Without PERFORM and CALL, the WHENEVER declarative is inadequate; the problem with supporting only CONTINUE and GO TO is that a program must pass control to a named procedure. The assumption is that an error is reported by the procedure, and the application halts execution. However, in many situations, an application can recover from errors. Therefore, your application should execute an error-handling procedure and pass control back to the main program.

For information about handling errors, see Diagnostics Area, next.

In NonStop SQL/MP, the WHENEVER declarative requires a colon (:) before the target location for the GO TO, PERFORM (in COBOL) or CALL (in C) actions. This restriction is removed in NonStop SQL/MX.

Diagnostics Area

NonStop SQL/MP provides a communication area (SQLCA) that enables an application program to check the status of SQL statement execution. You can declare the communication area by using the INCLUDE SQLCA directive.

NonStop SQL/MX does not provide the SQL communication area or use the INCLUDE SQLCA directive. Instead, NonStop SQL/MX follows the ANSI SQL:1999 standard by automatically providing a diagnostics area that stores completion and exception information.

At the beginning of the execution of an SQL statement, the diagnostics area is emptied by the preprocessor. When the statement executes, NonStop SQL/MX places completion and exception information in this area. You can access the information in the diagnostics area by using the GET DIAGNOSTICS statement.

Suppose your application has executed an SQL statement and the WHENEVER statement reports an error. To find out more about the results of the executed statements, code your application to execute a procedure that uses GET DIAGNOSTICS:

For example, in a C program, the code might look like this:

```
/* First, get the count of the number of exception conditions.
*/
EXEC SQL GET DIAGNOSTICS :hv_num = NUMBER,
                         :hv_cmdfcn = COMMAND_FUNCTION,
                         :hv_dynfcn = DYNAMIC_FUNCTION,
                          ...;
/* Second, get the i-th condition and process it. */
for (i = 1; i <= hv_num; i++) {
  EXEC SQL GET DIAGNOSTICS EXCEPTION :i
                         :hv_tabname = TABLE_NAME,
                         :hv_colname = COLUMN_NAME,
                         :hv_sqlstate = RETURNED_SQLSTATE
                          ...;
  ... /* Process the SQL error information. */
}
```

Statistics Area

NonStop SQL/MP provides a performance and statistics area (SQLSA) that enables an application to access this type of information about DELETE, INSERT, SELECT, UPDATE, OPEN, FETCH, DESCRIBE, and PREPARE statements. NonStop SQL/MX currently has no structure that corresponds to SQLSA and cannot provide information on what type of statement was compiled or on the cost of a PREPARE statement.

Program Development

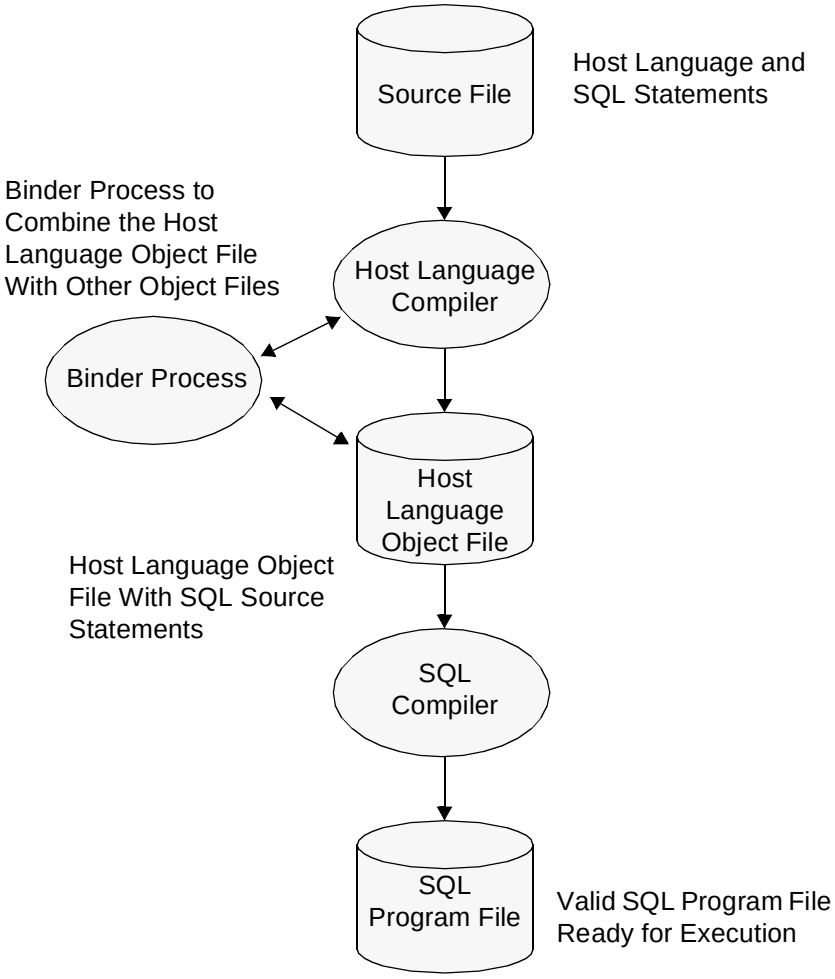
SQL/MP Compiler

Compiling programs using the SQL/MP compiler differs from using the SQL/MX compiler. In NonStop SQL/MP, compiling a host language program consists of these steps:

1. Create the host language source file with embedded SQL statements and directives.
2. Run the C or COBOL compiler to compile the statements in the source file.
3. Run the Binder program if necessary to bind object files.
4. Run the SQL compiler (SQLCOMP).

The result is a valid SQL program file, registered in the catalog and ready for execution. Graphically, this procedure appears similar to Figure 4-1.

Figure 4-1. SQL/MP Compilation Process



SQL/MX Compiler

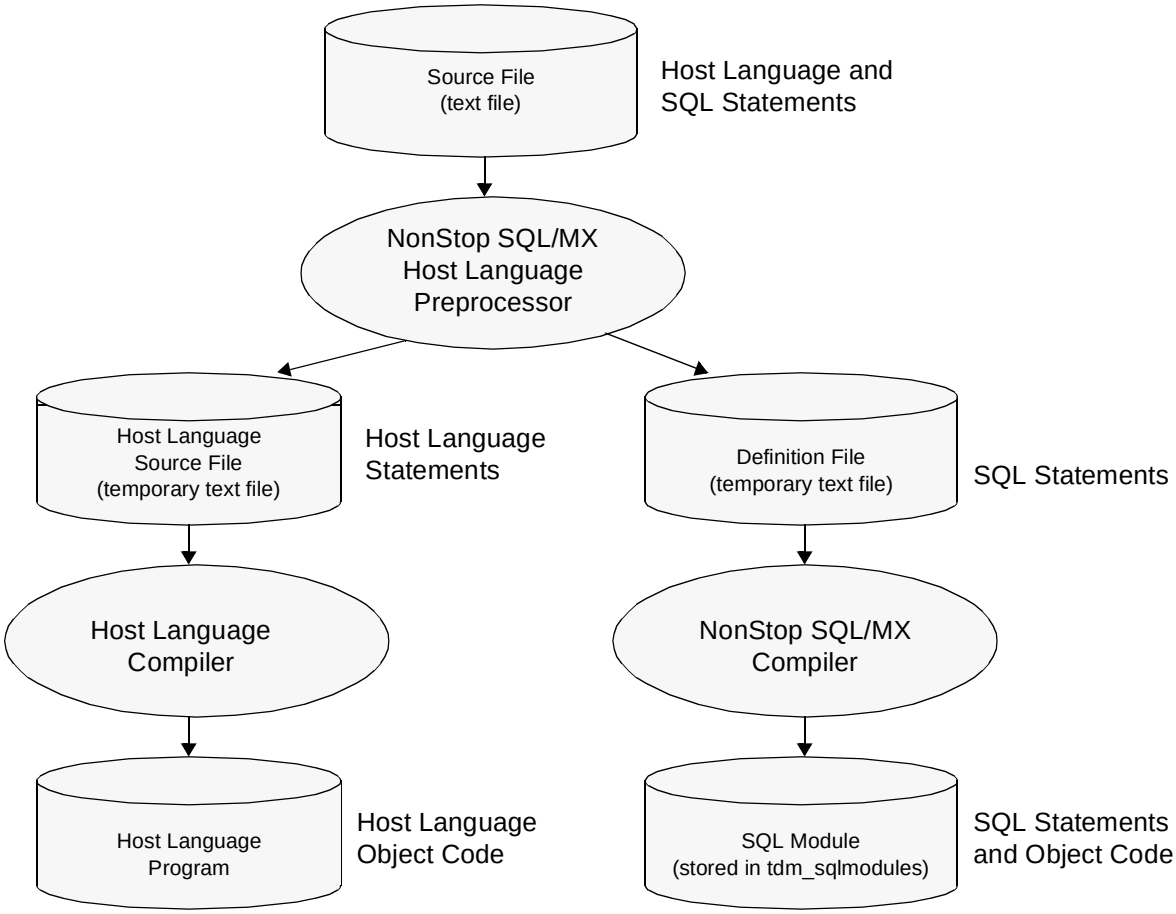
The SQL/MX program compilation process is different from that of NonStop SQL/MP. After you have compiled your program, you have two files: a host language source file and an SQL/MX module.

This high-level procedure compiles a program with NonStop SQL/MX and is shown in Figure 4-2 on page 4-12:

1. Using an editor, create the embedded SQL application.
2. Run the SQL/MX host language (C or COBOL) preprocessor to:
   - Parse the EXEC SQL statements and replace them with CLI calls.
   - Create a module definition file describing the SQL statements. The module definition file is a separate file instead of being embedded within the compiled objects.

3. Run the SQL/MX compiler to create execution plans for the SQL statements and store the plans in a module file.
4. Run the C or COBOL compiler to compile the statements in the source file.
5. Execute the program in the OSS environment, as described in the *SQL/MX Programming Guide for C and COBOL*.

Figure 4-2. SQL/MX Compilation Process



The key difference in the SQL/MX compilation is the resulting two files: the SQL module, which stores the query plan, and the host language program file.

For more information about the SQL/MX module, see the *SQL/MX Programming Guide for C and COBOL*. For syntax for the MODULE directive, see the *SQL/MX Reference Manual.*

Stored Procedures in Java

Both NonStop SQL/MP and NonStop SQL/MX allow you to use stored procedures, with these differences:

SQL/MP Stored Procedures

- Only callable from ODBC/MP.
- The called stored procedure must be a Pathway server with embedded SQL, written in COBOL, C, or TAL.
- Cannot use SQL/MX features like Publish/Subscribe.

SQL/MX Stored Procedures

- Callable from any SQL/MX application: MXCS, JDBC/MX, NonStop SQLJ, MXCI, or embedded SQL(MX) within a COBOL or C/C++ program.
- The called stored procedure must be written in Java.
- Implementation closely follows ANSI standards (SQL99 and SQL/JRT).

For more details on SQL/MX stored procedures, see the *SQL/MX Guide to Stored Procedures in Java*.

Similarity Checks and Automatic Recompilation

NonStop SQL/MX and SQL/MP perform similarity checking in different ways. These are differences between the products:

Difference between compile and run time module	Does SQL/MP similarity check pass?	Does SQL/MX similarity check pass?
Name of table	Yes	Yes
Data in the table	Yes	Yes
Number of partitions	Yes for serial plans No for parallel plans	No, unless plan chooses online transaction optimization
Partition key ranges	Yes for serial plans No for parallel plans	Yes
Non-partitioned table at compile time is partitioned at run time	Yes	No, unless plan chooses online transaction optimization
ESP Parallelism used in the plan	No	No

Difference between compile and run time module	Does SQL/MP similarity check pass?	Does SQL/MX similarity check pass?
Number of indexes - more indexes at run time than at compile time	Yes	No for INSERT, UPDATE, DELETE queries. Yes for SELECT.
Number of indexes - fewer indexes at run time than at compile time	Yes, if index is not used by plan	No for INSERT, UPDATE, delete queries. Yes, for SELECT, if index is not used by plan.
Unique index present both at compile and run time	Yes	No, fails for SELECT
Key tags (or values) for indexes	Yes	No
Creating and redefinition timestamp	Yes	Yes
Audit attribute	Yes, with minor exception	No
Any of these attributes: ALLOCATE, AUDITCOMPRESS, BUFFERED, CLEARONPURGE, EXTENT (primary and secondary), LOCKLENGTH, MAXEXTENTS, NOPURGEUNTIL, OWNER, SECURE, SERIALWRITES, TABLECODE, VERIFIEDWRITES	Yes	Yes
Statistics on the table	Yes	Yes
Column headings	Yes	Yes
Catalog where table is registered	Yes	Yes, could be in different schema
Number of columns - more columns at run time that at compile time	Yes, with same attributes and no SELECT	No
VIEWs on the table	Yes, only for qualifying protection views	No
Check constraints, referential integrity constraints, or triggers on the table	Yes. Ignores check constraints	No

5 Optimizer and Executor

Both the compiler and the executor for NonStop SQL/MX are different from NonStop SQL/MP. Some highlights of the new compiler include:

- Rule and cost-based optimizer
- Optimization of complex relational expressions
- Use of histogram statistics
- Extensible cost model

Some highlights of the rewritten executor include:

- Data-flow, scheduler-driven task model
- Increased parallelism

Optimizer

The optimizer is the stage in the SQL compiler that optimizes the query and determines the optimal query execution plan. Many design differences in NonStop SQL/MX occur before optimization. During the normalization phase, these tasks are performed:

- Subquery transformation into joins
- Constant folding (for example, `b < 0 * c` becomes `b < 0`)
- Predicates pushed down or pulled up
- Other unconditional transformations (for example, left joins transformed to inner joins)

This table highlights the differences between the SQL/MP and SQL/MX optimizers:

SQL/MP Optimizer

Search engine uses dynamic programming (bottom-up optimization) to enumerate possible join orders. Considers Cartesian products as part of the join enumeration.

Optimizations considered include:

- Three-join algorithms (nested join, hybrid hash join, and merge join).
- Parallel join algorithm (matching, nonmatching).
- Repartitioned join.
- Aggregation and grouping in DAM (aggregate and sort GROUP BY).
- Unions cannot be parallelized.
- Transitive closure; equality predicates are rewritten to include implied predicates.

SQL/MX Optimizer

Search engine uses branch and bound programming (top-down optimization). Multipass design; in the pilot phase, tables are ordered in ascending order, and large tables are scanned last.

Optimizations considered include:

- Four-join algorithms (nested join, ordered hash join, hybrid hash join, and merge join). Within a join, also supports range repartitioning.
- Extensible query tree allows for parallel operations anywhere in the query tree.
- Repartitioned join.
- Aggregation and grouping can be executed in DAM and ESPs.
- Unions can be parallelized.
- Transitive closure more extensive; extends to grouping columns, order by columns, and column references in the select list.
- GROUP BY operations can occur almost anywhere in the query tree.

SQL/MP Optimizer

Costing and statistics:

- Uses captured statistics to estimate the number of rows. Consists of a number of values—UEC, second high and second low values—on a per-column basis.
- Numeric data requires data be uniformly distributed (between the second high and second low values).
- Costing based on simple arithmetic.
- Cost is represented by a single floating point value.

Influencing query plan:

- Force plans through the CONTROL directive (table scope).
- Change a few defaults with CONTROL directives on a table basis.
- Update statistics.
- Modify the table, such as adding indexes.

Transformations within queries:

- User must know how the optimizer handles predicates so that proper query design yields an optimal plan.
- Some predicate transformations done.
- Predicates are pushed down as far as possible.

SQL/MX Optimizer

Costing and statistics:

- Histograms are used to represent data distribution, not just second high and second low values. Data distribution is divided into intervals. For each interval, NonStop SQL/MX stores the number of values and the UEC.
- Multicolumn UEC and row count.
- Histograms can capture the true distribution, given enough histogram intervals.
- Costing aligned to concurrency and parallelism.
- Cost is represented as cost vectors composed of resource units.

Influencing query plan:

- Force plans through the CONTROL QUERY SHAPE statement (query tree scope).
- Change user defaults in several ways: per query tree, table basis, system or session basis.
- Update statistics.

Optimizer internally handles transformations within queries. Some transformations include:

- Unconditional predicate transformations (for example, constant folding).
- Subqueries transformed to joins.
- Predicates rewritten and pushed down as far as possible (more than in NonStop SQL/MP).
- Syntactic and semantic sort elimination.

Executor

The executor in NonStop SQL/MX has also been rewritten and takes advantage of these features:

- Completely no-waited operations
- More work done in the DAM
- Multiple forms of parallelism
- Scalability through parallelism on multiple levels

This table highlights the differences between the SQL/MP executor and the SQL/MX executor.

SQL/MP Executor	SQL/MX Executor
Control flow iterator model. See SQL/MP Iterator Model.	Data-flow, scheduler-driven task model. See SQL/MX Task Model.
Nested join, hash join, and merge join supported.	Nested join, hash join, and merge join supported. Within a join, also supports range partitioning.
Aggregate and sort GROUP BY queries can be executed in DAM.	Aggregate, sort and hash GROUP BY queries can be executed in DAM.
Parallel join, GROUP BY, sort operations supported through ESPs.	Increased parallelism supported through no-wait interface to DAM (no ESPs needed for some parallel plans).
Result data from DAM and inserted rows can be buffered in larger messages (VSBB, RSBB, and Insert VSBB).	Request and result data to and from DAM and inserted rows are always buffered.
Uses ESPs to keep multiple DAM processes active at the same time. Requires a 1:1 mapping between ESPs and DAMs of a partitioned table.	Does not require 1:1 mapping between ESPs and DAMs of a partitioned table.

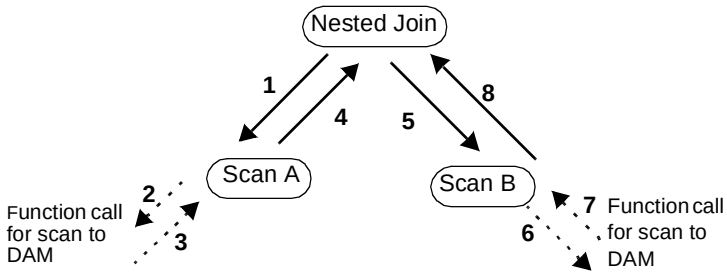
SQL/MP Iterator Model

The SQL/MP executor uses an iterator model, also called a *control flow model*. In the iterator model, a query plan is executed by a series of function calls. Each function implements an operator in the query plan. Because the implementation is through a series of function calls, no pipelining or parallel processing is possible.

The interface between the SQL/MP executor and DAM is synchronous. The executor process is blocked while it waits for a reply from a DAM. NonStop SQL/MP uses ESPs to keep multiple DAMs active at the same time. NonStop SQL/MP requires a 1:1 mapping between ESPs and the DAMs of a partitioned table.

The next figure shows an example of the iterator model. This iterator model is a serialized type of model, meaning that only one function is active at a time. The numbers in the figure relate to these operations:

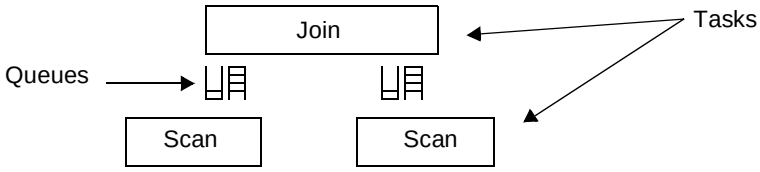
1. The join calls the scan function.
2. The scan asks DAM to retrieve rows.
3. Rows are returned from DAM to the scan.
4. Rows are returned from the scan function call to the join.
5. The join calls the scan function, providing as input the row produced by Step 4.
6. The scan asks DAM to retrieve all rows matching the input.
7. Rows are returned from DAM to the scan.
8. Rows are returned from the scan function to the join.



SQL/MX Task Model

NonStop SQL/MX uses a data flow and scheduler-driven task model to execute queries. After a query is optimized, the optimizer generates an optimized, executable query plan that goes to the executor. The executor creates an operator tree based on the executable query plan and prepares a node for each operator.

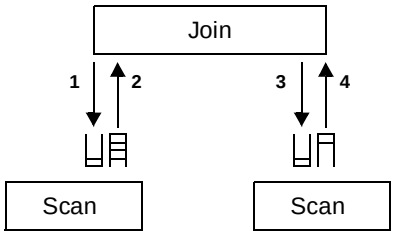
The next figure shows an operator tree in the task model. Each operator is an independent task, and data flows between operators through in-memory queues or by interprocess communication. The queue pair operates between two operators. Queues between tasks allow operators to exchange multiple requests or result rows at a time. A scheduler coordinates the execution of tasks and runs whenever it has data on one of its input queues.



Building on the previous figure, the next figure shows the steps that the executor might take to process this simplified query tree. The numbers in the figure relate to the operations:

1. The join places a request in the down queue of the left scan.
2. The scan places a group of rows in its up queue, which the join retrieves.
3. The join places a request in the down queue of the right scan. At the same time, the join can place a request in the down queue of the left scan.
4. The right scan places a group of rows in its up queue, which the join retrieves. If active, the left scan also places a group of rows in its up queue, which the join retrieves.

The initial request from the join to the left scan queue must occur before the other operations can begin. After the initial request is processed, both the left and right scans can be processing information through their queues simultaneously.



The task model makes it easy to perform all internal operations asynchronously so that a single server thread can have multiple I/Os outstanding. This model also allows parallelism for both shared memory and distributed memory architectures. In-memory queues are used for communication, while exchange operators are used for distributed memory.

Parallelism

In NonStop SQL/MP, parallelism works only on a table that is already partitioned and only for queries that retrieve data from multiple partitions. NonStop SQL/MX considers partitioned parallelism in all cases. In addition, NonStop SQL/MX uses pipelined and independent parallelism.

NonStop SQL/MP does not recognize multilevel parallelism. In NonStop SQL/MP, either the entire query tree runs in parallel or none of it runs in parallel. NonStop SQL/MX supports multilevel parallelism, which allows each operator to have a different level or degree of parallelism, including none.

NonStop SQL/MP uses only ESP parallelism and needs one ESP per base table partition. The number of ESPs in NonStop SQL/MX is determined more by the number of available CPUs than it is by the number of partitions in the tables. If a table has more partitions than available CPUs, the optimizer can group the partitions so that each ESP processes multiple partitions as if they were a single partition. Each ESP will group adjacent partitions.

NonStop SQL/MX can merge sorted parallel streams with some restrictions. For example, a parallel merge join with an ORDER BY clause might not require a final sort. In SQL/MP, multiple streams must be sorted.

For a full explanation of parallelism in NonStop SQL/MX, see the *SQL/MX Query Guide*.

Ways to Influence the Query Execution Plan

In NonStop SQL/MP, you can write queries in different ways to affect how a query is optimized. You must know how the SQL/MP optimizer handles predicates so that a proper query design yields an optimal plan. You can write queries in such a way as to eliminate certain types of joins. Query performance in NonStop SQL/MX is not affected by query design, because the SQL compiler performs the majority of transformations internally.

This list indicates a few actions you can take to influence a query execution plan in NonStop SQL/MX:

- Setting default values. See Default Settings.
- Updating statistics regularly.
- Forcing query execution plans with the CONTROL QUERY SHAPE statement.

For detailed information about improving query performance, setting default values, updating statistics, and forcing query execution plans, see the *SQL/MX Query Guide*.

Default Settings

In NonStop SQL/MP, you specify certain default settings through the CONTROL directives. For example, the CONTROL EXECUTOR directive allows you to specify whether you want parallel processing on or off. With the CONTROL TABLE directive, you can specify that you want to use a certain join method or change the timeout setting. With the CONTROL QUERY directive, you can adjust other default values.

In NonStop SQL/MX, the SYSTEM_DEFAULTS table is a user table that enables you to specify default settings for options and other attributes when executing MXCI commands and SQL queries that can be run through MXCI or within an application. For some attributes, you can override a default value by specifying an option with the statement or command. You might find it more efficient, however, to configure the SQL/MX execution environment with default values. NonStop SQL/MX lets you change the default settings of many more items than NonStop SQL/MP.

If you do not enter an option when entering a command or if you do not provide some attribute associated with the execution of queries, NonStop SQL/MX chooses a default. If you have specified system default settings for the execution environment, these value are used. Otherwise, the SQL/MX default values are used.

In addition to configuring the execution environment with default values through the defaults tables, in NonStop SQL/MX you can use the CONTROL TABLE and CONTROL QUERY DEFAULT settings to change default values at run time. For more information about these commands, see Section 3, DML Features and the *SQL/MX Reference Manual*.

NonStop SQL/MX stores system defaults information in the SQL/MP user table named DEFAULTS, which is registered in the system catalog. For a full listing of default values that can be changed, see the SYSTEM_DEFAULTS Table entry in the *SQL/MX Reference Manual*. See also the *SQL/MX Query Guide*.

This table lists some of the differences in default values between NonStop SQL/MP and NonStop SQL/MX. If you want to mimic an SQL/MP system using NonStop SQL/MX, you need to reset some default values to get the NonStop SQL/MP behavior.

Default	NonStop SQL/MP	NonStop SQL/MX
ISOLATION_LEVEL	STABLE ACCESS	READ_COMMITTED
PARALLEL_EXECUTION	OFF	ON
READ_ONLY CURSOR	ON	OFF
RECOMPILATION_WARNINGS	ON	OFF
MDAM	OFF	ON

EXPLAIN Features

Both NonStop SQL/MP and NonStop SQL/MX utilize EXPLAIN features to present information about query execution plans. In SQL/MP, EXPLAIN is a directive or SQL utility that describes the execution plan for queries.

In NonStop SQL/MX, you review query execution plans and query performance by using the EXPLAIN function, DISPLAY_EXPLAIN command, and Visual Query Planner application. The EXPLAIN function and DISPLAY_EXPLAIN command present query execution plans in machine-readable format. The Visual Query Planner presents query execution plans in a graphical user interface through MXCS. Detailed information about each method is presented in the *SQL/MX Query Guide*.

Forcing Query Execution Plans

In NonStop SQL/MP, you force query execution plans with options on the CONTROL TABLE directive on a per table basis. In NonStop SQL/MX, you force query execution plans by using the CONTROL QUERY SHAPE statement to force the entire plan, not just a particular table.

The EXPLAIN function and the CONTROL QUERY SHAPE statement use similar identifiers for the nodes of an operator tree. You can also generate and force query execution plans through the Visual Query Planner.

Usually, the SQL/MX optimizer presents the best query execution plan. On rare occasions, you might want to revise the plan presented by the optimizer. Forcing the plan means that you take control for your query execution plan away from the optimizer.

To force a plan, you must first run the EXPLAIN function to generate the result table of the EXPLAIN function for your DML statement. Next, review the result to see if the plan you received was the plan you wanted. If you determine that you want to change the plan, use the CONTROL QUERY SHAPE statement to force the plan.

For additional information about forcing query execution plans, see the *SQL/MX Query Guide*.

Statistics

In NonStop SQL/MP, the UPDATE STATISTICS statement updates the statistics stored in the catalog for the specified table. In NonStop SQL/MX, the UPDATE STATISTICS statement updates the histogram statistics for a group of columns within a table.

NonStop SQL/MX provides a method for generating histograms that shows how data is distributed for a column within a table. The purpose of generating these statistics is to enable the optimizer to create more efficient data query execution plans.

When generating a histogram for a table, NonStop SQL/MX divides the range of specified column values of the table into some number of intervals, distributing the rows evenly within these intervals. It computes statistics associated with each interval and uses the statistics to devise optimized plans.

For further information, see the UPDATE STATISTICS Statement and User Metadata (UMD)—Histogram Tables entries in the *SQL/MX Reference Manual*. The *SQL/MX Query Guide* contains a section on keeping statistics current and best practices to follow with respect to statistics.

6

Data Definition Language (DDL) Differences

With NonStop SQL/MX Release 2, NonStop SQL/MX can now create objects. Prior to Release 2, NonStop SQL/MX did not allow creation of objects. NonStop SQL/MX can access NonStop SQL/MX as well as SQL/MP files. New Data Definition Language (DDL) statements allow you to create and modify application files. For details on these statements, see “SQL/MX Statements” in the *SQL/MX Reference Manual*.

SQL Statements

You can now create, alter, and drop SQL/MX database objects similarly to how you manipulate these objects in NonStop SQL/MP, with these exceptions:

- You cannot create collations for SQL/MX tables, so there is no CREATE COLLATION statement. You create constraints at CREATE TABLE time or later with a ALTER TABLE ADD CONSTRAINT statement, so there is no separate CREATE CONSTRAINT statement. You drop constraints with the ALTER TABLE DROP CONSTRAINT statement.
- You cannot alter catalogs, collations, programs or views.
- The generic DROP command has been replaced by specific DROP CATALOG, DROP INDEX, DROP PROCEDURE, DROP SCHEMA, DROP SQL, DROP SQLMP ALIAS, DROP TABLE, DROP TRIGGER, and DROP VIEW statements.

See the *SQL/MX Reference Manual* for information about statements and database objects.

Database Object Names

You must use ANSI logical names (a three-part logical name: catalog-name. schema-name.object-name) to describe SQL/MX objects, using the LOCATION clause to specify their physical location.

You can use physical names, ANSI logical names, SQL/MP alias names, or DEFINE names to refer to SQL/MP objects.

See “SQL/MX Language Elements” in the *SQL/MX Reference Manual* for details about database object names.

ANSI Names

Namespace for SQL/MX objects is organized in a hierarchical manner. Database objects exist in schemas, which are themselves contained in catalogs. Catalogs are

collections of schemas. Schema names must be unique within a given catalog. Multiple objects with the same name can exist provided that each belongs to a different namespace. NonStop SQL/MX supports various types of namespaces. See “SQL/MX Language Elements” in the *SQL/MX Reference Manual* for details about these namespaces.

You reference SQL/MP database objects by using their Guardian physical names. To allow for the use of ANSI logical names, NonStop SQL/MX creates a metadata table, the OBJECTS table. This table is used to store object names. Mappings from logical object names to physical Guardian locations are stored in the metadata table MP_PARTITIONS.

Use the CREATE SQLMP ALIAS command within your application to create needed mappings from logical to physical names. This command inserts a mapping as a row in the OBJECTS table. SQL/MP aliases are true ANSI names that represent the underlying Guardian physical names of SQL/MP objects, in which the catalog part refers to an existing catalog and the schema part refers to an existing schema.

Organization of Catalogs and Schemas

NonStop SQL/MX and NonStop SQL/MP have different concepts of catalogs.

SQL/MX Catalogs

An SQL/MX catalog is a named logical object that contains descriptions of a set of schemas. You can access SQL/MX objects with the three-part name of the actual object.

A catalog can contain multiple schemas, each possibly owned by a different user. A catalog cannot contain other catalogs. Any user on a node can create a catalog on that node. Any user on a node can drop an empty catalog on that node. For more information, see “SQL/MX Language Elements” in the *SQL/MX Reference Manual*.

NonStop SQL/MX uses ANSI-compliant CREATE and DROP statements to create and drop catalogs and schemas. Note that catalog names beginning with “NONSTOP_SQLMX_” are reserved for system metadata.

SQL/MP Catalogs

An SQL/MP catalog is a set of tables and indexes that describe SQL objects. Tables in the set are called catalog tables and SQL creates them, along with their indexes, when you execute a CREATE CATALOG statement. Each SQL/MP catalog (the set of catalog tables and their indexes) resides on its own Guardian subvolume, and the name of that subvolume is also the name of the catalog.

Each node on which SQL/MP is used has one special catalog called the system catalog and might have many other catalogs. Each table, view, index, partition, collation, or catalog table located on a node must be described in a catalog on the

same node. Normally, an SQL program is registered in a catalog, too. For more information, see the *SQL/MP Reference Manual*.

NonStop SQL/MP allows you to create catalogs and to drop system catalogs.

Metadata

The system catalog is NONSTOP_SQLMX_*nodename*. SQL/MX metadata is organized within these system schemas:

SYSTEM_SCHEMA	System schema tables that list information such as catalogs and schemas in the system.
DEFINITION_SCHEMA_ VERSION_*versnum*	Definition schema tables that list information such as columns in tables, constraints, partitions, views, procedures and so on.
SYSTEM_DEFAULTS_ SCHEMA	User-provided system defaults.
MXCS_SCHEMA	MXCS subsystem data source and governing information.
SYSTEM_SQLJ_SCHEMA	System schema that contains only the stored procedure VALIDATEROUTINE, which is for internal use.

SQL/MX's metadata is on the same subvolume as the system catalog, which you specify when you install NonStop SQL/MX.

SQL/MP metadata, also called the Data Dictionary, is organized by disk. It consists of all the catalogs on a network and labels for each file on the volume. The label includes the name or the object, the name of the catalog that describes the object, and other information about the file.

SQL/MP metadata is the combination of system catalog metadata and user catalog metadata. NonStop SQL/MP creates the system catalog when you issue the CREATE SYSTEM CATALOG command, in whatever catalog you specify. The default is $SYSTEM.SQL. Each catalog of user metadata resides on its own Guardian subvolume.

Previous releases of NonStop SQL/MX use a different table scheme for metadata where metadata resided in SQL/MP tables registered in the SQL/MP catalog. For SQL/MX Release 2.0, metadata resides in SQL/MX metadata tables. The MIGRATE utility is provided to move metadata from previous releases of NonStop SQL/MX to current metadata tables. For more information about this utility, see the *SQL/MX Installation and Management Guide* and the *SQL/MX Reference Manual*.

Tables

You create constraints on an SQL/MX table with the CREATE TABLE and ALTER TABLE ADD CONSTRAINT statements. You create constraints on SQL/MP tables with the CREATE CONSTRAINT statement.

Clustering Keys

You assign a clustering key for an SQL/MX table by using the STORE BY clause of the CREATE TABLE statement. If you enter STORE BY PRIMARY KEY, NonStop SQL/MX bases the clustering key on the primary key. The STORE BY clause determines the order of rows in the physical file and affects how you can partition the file.

You assign a clustering key for an SQL/MP table by using the CLUSTERING KEY clause of the CREATE TABLE statement, listing the columns that will be used for this key. SQL/MP stores rows in ascending or descending order, as you specify. You cannot control the order of rows in the physical file.

Table Attributes

NonStop SQL/MX allows you to set these table attributes: ALLOCATE, AUDITCOMPRESS, BLOCKSIZE, CLEARONPURGE, EXTENT, and MAXEXTENTS.

NonStop SQL/MP allows you to set these table attributes: ALLOCATE, AUDIT, AUDITCOMPRESS, BLOCKSIZE, BUFFERED, CLEARONPURGE, DCOMPRESS, EXTENT, FORMAT, ICOMPRESS, LOCKLENGTH, MAXEXTENTS, NOPURGEUNTIL, RECLENGTH, TABLECODE, and VERIFIEDWRITES.

File Types

You can access these type of SQL/MP files through NonStop SQL/MX:

- Key-sequenced tables with or without partitions
- Entry-sequenced tables that are not partitioned

You cannot access these type of SQL/MP files through NonStop SQL/MX:

- Entry-sequenced tables that are partitioned
- Relative tables

Constraints

You create constraints on SQL/MX tables with the CREATE TABLE statement, and you can add or drop constraints with the ALTER TABLE statement. For more information, see “SQL/MX Language Elements” in the *SQL/MX Reference Manual*.

To create constraints on an SQL/MP table, use the SQL/MP CREATE CONSTRAINT statement when you create the table. To drop constraints on an SQL/MP table, use the SQL/MP DROP statement.

Indexes

You can create indexes for SQL/MX files, with these differences from NonStop SQL/MP:

- Index table attribute

  NonStop SQL/MX allows you to set these index table attributes: ALLOCATE, AUDITCOMPRESS, BLOCKSIZE, CLEARONPURGE, EXTENT, and MAXEXTENTS.

  NonStop SQL/MP allows you to set these index table attributes: ALLOCATE, AUDITCOMPRESS, BLOCKSIZE, BUFFERED, CLEARONPURGE, DCOMPRESS, EXTENT, FORMAT, ICOMPRESS, ISLACK, LOCKLENGTH, MAXEXTENTS, NOPURGEUNTIL, SERIALWRITES, SLACK,TABLECODE, and VERIFIEDWRITES.

- Index table access

  NonStop SQL/MX does not support WITH SHARED ACCESS.

- Partition changes

  NonStop SQL/MX allows you to define secondary partitions as range or hash partitions.

- Populating indexes

  If you are creating an index on a large SQL/MX table that is already populated, you should use the NO POPULATE option, and then run the POPULATE INDEX utility to load the index. Because CREATE INDEX executes in a single TMF transaction, it could experience TMF limitations such as a transaction timeout if a large amount of data is to be moved. See the *SQL/MX Installation and Management Guide* for information about creating and populating indexes and the *SQL/MX Reference Manual* for details about the CREATE INDEX statement and the POPULATE INDEX utility.

- PARALLEL EXECUTION ON | OFF option

  NonStop SQL/MX uses the optimizer to determine whether to perform parallel index loading. It is not a specifiable attribute.

- INVALIDATE | NO INVALIDATE option

  SQL./MX does not allow you to control whether or not changes to indexes cause similarity checks and possible recompiles.

Partitions

NonStop SQL/MX allows you to locate as many partitions as you want on the same disk. Like SQL/MX tables, partitions use ANSI logical names. You can specify Guardian names for a partition.

NonStop SQL/MP allows no more than one partition of an object on each disk. SQL/MP partitions have Guardian names.

You manage partitions differently in NonStop SQL/MX and NonStop SQL/MP. For details, see Partition Management: The MODIFY Utility on page 7-2

Views

SQL/MX Views

The distinction between protection and shorthand views does not exist for SQL/MX views. To create a view, you must have SELECT privileges for the objects underlying the view.

NonStop SQL/MX supports UNION in a view definition.

For information about SQL/MX views, see “SQL/MX Language Elements” in the *SQL/MX Reference Manual*.

SQL/MP Views

An SQL/MP view is either a protection view or a shorthand view. A protection view is derived from a single table and can be read, updated, and secured. A shorthand view is derived from one or more tables or other views and inherits the security of the underlying tables. A shorthand view can be read but not updated.

To create a view, you must have authority to write to the catalog that receives the view description and to the USAGES tables of catalogs describing the underlying tables and views.

To create a protection view, you must also be a generalized owner of the underlying table. Any partitions or indexes of the table underlying the protection view must be accessible when you create the view. To specify read or write access for a protection view, you must have authority to read or write to the underlying table and all associated indexes (unless you are the super ID).

NonStop SQL/MX and NonStop SQL/MP handle security on views differently. In NonStop SQL/MP, you need to have appropriate security on the underlying tables of the view in order to update the view. In NonStop SQL/MX you need only to have appropriate security on view itself. You do not need to have privileges on the underlying table.

For information about SQL/MP views, see Views in the *SQL/MP Reference Manual*.

Object Security

NonStop SQL/MX uses GRANT and REVOKE statements that conform to the ANSI SQL standard to control privileges for objects. It does not support a SECURE clause. Security is controlled internally by the software and does not use the security vector.

NonStop SQL/MP uses the SECURE clause to specify a Guardian security string, “RWEP” to secure objects. security is controlled by the Guardian security vector stored in the label.

7 Utility Differences

SQL/MX 2.0 includes utilities that perform tasks such as populating indexes, displaying information about files or DDL, managing partitions, and so on. These SQL/MX utilities are similar to those in NonStop SQL/MP, but not identical:

DISPLAY USE OF Command

In NonStop SQL/MX, DISPLAY USE OF is an MXCI command that provides usage information only for statically compiled modules, such as:

- Modules using a specific SQL object such as a table, index, or MP alias
- All the SQL objects used by a module or set of modules

NonStop SQL/MP's implementation of this command provides information on various objects:

- All objects using a specific SQL object, such as a table, index, view, or collation. NonStop SQL/MP displays all tables, views, indexes, collations, and programs associated with that object.
- Optionally, detailed information on each object displayed.

For syntax and description of this command, see "MXCI Commands" in the *SQL/MX Reference Manual*.

IMPORT Command

You can use SQL/MX's IMPORT command to import data from a file (ASCII or binary) into a NonStop SQL/MX table. IMPORT executes at the OSS or MXCI command prompt. It is similar to but not equivalent to SQL/MP's LOAD command. See "SQL/MX Utilities" in the *SQL/MX Reference Manual* for details.

MXTOOL Utility

Earlier releases of NonStop SQL/MX supported only a limited set of tools with the MXTOOL utility. NonStop SQL/MX Release 2 provides a tool, the MXTOOL utility, that gives you additional options. The MXTOOl utility allows you verify the contents of an SQL/MX object, fix parts of the metadata and label information, return information about SQL/MX objects, and remove both damaged and undamaged objects from metadata or labels in a uniform way.

The MXTOOL utility performs these operations:

FIXUP Similar to various SQL/MP operations

GOAWAY	Similar to SQL/MP's GOAWAY command
INFO	Displays ANSI names and versions of Guardian files used by an object
VERIFY	Similar to SQL/MP's VERIFY command

For more information, see "SQL/MX Utilities" in the *SQL/MX Reference Manual*.

Partition Management: The MODIFY Utility

In both NonStop SQL/MX and NonStop SQL/MP, you create partitioned tables and indexes with CREATE TABLE and CREATE INDEX.

In NonStop SQL/MX you use the MODIFY utility to manage partitions. You can:

- Reuse an existing partition of a range-partitioned table
- Add, drop, or move partitions of range-partitioned tables and indexes
- Add, drop, or move partitions of hash-partitioned tables and indexes
- Move partitions of system-clustered objects

In NonStop SQL/MP, you use the PARTONLY MOVE clause of ALTER TABLE (for a key-sequenced file) to break the table into partitions or to break a partition into additional partitions, or the ADD PARTITION clause of ALTER TABLE to add a partition to the end of an entry-sequenced or relative table. To create a partitioned index, use CREATE INDEX or the PARTONLY MOVE clause of ALTER INDEX.

For a description of the MODIFY command, see "SQL/MX Utilities" in the *SQL/MX Reference Manual*.

POPULATE INDEX Command

NonStop SQL/MX includes a CREATE INDEX command, but it includes a NO POPULATE option you can use if you prefer to populate the index in a separate operation from creating it. For instance, if you are creating an index on a large table that is already populated, you should use the NO POPULATE option and then run the POPULATE INDEX utility to load the index. Because CREATE INDEX executes in a single TMF transaction, it could experience TMF limitations such as a transaction timeout if there is a large amount of data to be moved.

The SQL/MX statements CREATE INDEX NO POPULATE followed by POPULATE INDEX are equivalent to the SQL/MP CREATE INDEX statement.

For more information, see the *SQL/MX Reference Manual*.

SHOWDDL Command

SQL/MX's SHOWDDL command displays a DDL statement that would create a table, view, or stored procedure as it exists in metadata, including the object's dependent

objects. This output can be used to re-create the specified object, including its dependent objects.

SHOWDDL is similar to SQL/MP's INVOKE command, which produces a record description that corresponds to a row in a specified table or view.

See "MXCI Commands" in the *SQL/MX Reference Manual* for more information.

SHOWLABEL Command

SQL/MX's SHOWLABEL command displays file-label and resource-fork information for SQL/MX tables, worktables, views, and indexes. The information includes the object version, physical location, and other characteristics.

Every SQL object includes a logical file label to store the object's file attributes and information about its dependent objects. The resource fork is a new file that contains structural descriptions of a table. When an SQL/MX object is created, two physical files are instantiated: the data fork and the resource fork. The data fork is where the user data resides. The resource fork contains structural information, such as the partition map.

SHOWLABEL is similar to SQL/MP's FILEINFO utility, which displays information about the versions and physical characteristics of tables, indexes, views, collations, Enscribe files, and OSS files.

For more information, see "MXCI Commands" in the *SQL/MX Reference Manual*.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Special Characters