# HP NonStop SQL/MX Glossary


**Abstract**

This glossary describes terms that are specific to HP 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 |
|---|---|
| 523732-002 | June 2005 |

**Document History**

| Part Number | Product Version | Published |
|---|---|---|
| 520363-001 | NonStop SQL/MX Release 1.5 | November 2001 |
| 524766-001 | NonStop SQL/MX Release 1.5 | August 2002 |
| 523731-001 | NonStop SQL/MX Release 1.8 | December 2002 |
| 523732-001 | NonStop SQL/MX Release 2.0 | April 2004 |
| 523732-002 | NonStop SQL/MX Releases 2.0 and 2.1 | June 2005 |

# HP NonStop SQL/MX Glossary

Glossary

# Glossary (continued)

# What's New in This Manual

## Manual Information

### Abstract

This glossary describes terms that are specific to HP 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 |
|---|---|
| 523732-002 | June 2005 |

### Document History

| Part Number | Product Version | Published |
|---|---|---|
| 520363-001 | NonStop SQL/MX Release 1.5 | November 2001 |
| 524766-001 | NonStop SQL/MX Release 1.5 | August 2002 |
| 523731-001 | NonStop SQL/MX Release 1.8 | December 2002 |
| 523732-001 | NonStop SQL/MX Release 2.0 | April 2004 |
| 523732-002 | NonStop SQL/MX Releases 2.0 and 2.1 | June 2005 |

# New and Changed Information

These terms have been added or changed for this release:

| | | |
|---|---|---|
| alias mapping | MPALIAS table | SQL/MP alias |
| ANSI external name | OLT optimization | SQL/MX catalog manager |
| ANSI internal name | SQL-92 | user metadata tables |
| bulk move | SQL:1999 | |
| mixed-workload environment | SQL:2003 | |

The technical content of this publication has been updated and reflects the state of the product at the G06.26 RVU.

# About This Manual

This manual lists terms relevant to NonStop SQL/MX and related subsystems.

## Audience

This manual was written for SQL/MX database administrators and programmers who are using the SQL/MX conversational or programmatic interface.

Readers of this manual should know the fundamentals of:

- SQL/MX
- The HP NonStop operating system
- The SQL/MX conversational interface (MXCI)

## Related Documentation

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

**Introductory Guides**

| | |
|---|---|
| *SQL/MX Comparison Guide for SQL/MP Users* | Describes SQL differences between SQL/MP and 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 for, 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. |
| *NSM/web Installation Guide* | Describes how to install NSM/web and troubleshoot NSM/web installations. |

### 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

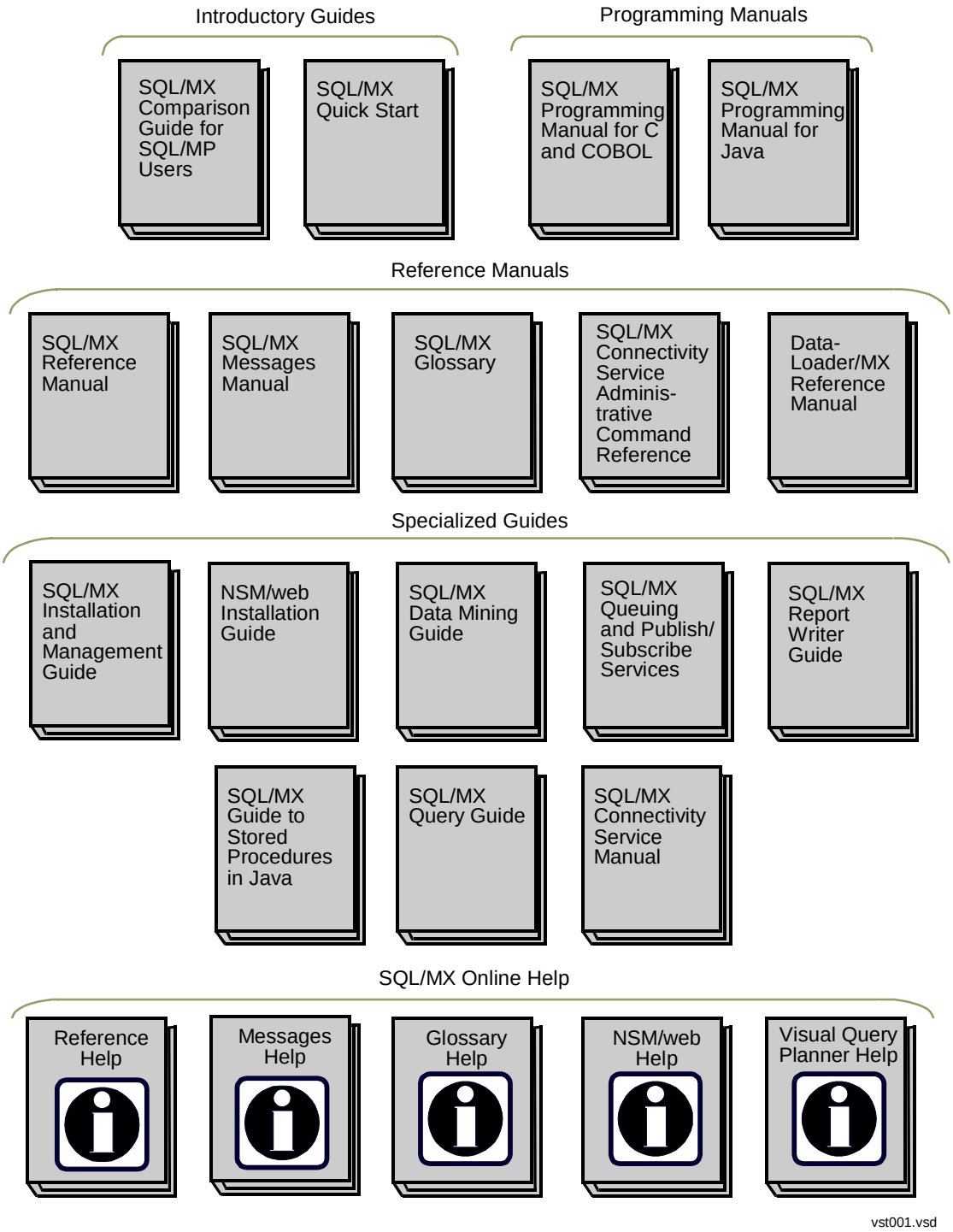
Technical Library (NTL). For more information about downloading online help, see the *SQL/MX Installation and Management Guide*.

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 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
```

## Change Bar Notation

Change bars are used to indicate substantive differences between this manual and its preceding version. Change bars are vertical rules placed in the right margin of changed portions of text, figures, tables, examples, and so on. Change bars highlight new or revised information. For example:

The message types specified in the REPORT clause are different in the COBOL85 environment and the Common Run-Time Environment (CRE).

The CRE has many new message types and some new message type codes for old message types. In the CRE, the message type SYSTEM includes all messages except LOGICAL-CLOSE and LOGICAL-OPEN.

# Glossary

A B C D E F G H I J K L

M N O P Q R S T U V W

## A

**3GL.** Third-generation language. A high-level programming language such as C or Java.

**4GL.** A nonprocedural, high-level language, such as structured query language (SQL), that is used for database management systems.

**aborted transaction.** A transaction that was backed out (undone). See also transaction

**access option.** A DML statement option that affects the consistency of data accessed by the statement execution. See also read uncommitted access, read committed access, serializable access, repeatable read access.

**access path.** The method by which data is accessed. SQL/MX provides primary access and alternate index access. Primary access can be sequential (entire base table scanned) or keyed (rows read based on clustering key value). Alternate index access can be index only (all columns required by the query are included in the index) or access to the base table through an index.

**access plan.** A database access method for a single compiled DML statement. The EXPLAIN function is a table-valued stored function that generates a result table describing an access plan for the statement. See also execution plan.

**alias mapping.** A mapping from a logical object name to a physical NonStop operating system location. Alias mappings allow three-part logical names to identify SQL/MP tables and views. In Release 1.8 and earlier releases, this mapping was stored in a special SQL/MP table created by SQL/MX called MPALIAS. Starting with Release 2.0, mappings are created in existing catalogs and schemas and are stored in SQL/MX metadata as SQL/MX aliases that consist of the catalog, schema, and name. See also object name and MPALIAS table.

**alternate key.** See index.

**anchor file.** A simple text file with ASCII tokens that identifies the volume where the system metadata resides. The anchor file resides in the NonStop operating system space `$SYSTEM.ZSQLMX.MXANCHOR`.

**ANSI external name.** A name defined when objects are created or altered. ANSI SQL objects have both external and internal names. When objects are created or altered in

SQL/MX, they are given external names. ANSI external names must conform to ANSI naming conventions. See also ANSI internal name.

**ANSI internal name.** A name defined when objects are stored in the metadata. ANSI SQL objects have both external and internal names. When objects are created or altered in SQL/MX, they are given external names. However, when the same objects are stored in the metadata, they are stored by their internal names. ANSI internal names must conform to the NonStop operating system naming format. See also ANSI external name.

**ANSI view.** A named specification of a result table, which is a set of rows, selected or generated from one or more base tables.

**application.** A computer program, such as a word processor or spreadsheet, designed to assist in the performance of a specific task.

**application programming interface (API).** A set of functions or procedures that an application uses to request and carry out lower-level services. For example, the ODBC API consists of the ODBC functions.

**approximate numeric data type.** A numeric data type that is an approximation of a real number and consists of two parts: a mantissa and an exponent. The mantissa is a signed numeric value, and the exponent is a signed integer that specifies the magnitude of the mantissa. The SQL approximate numeric types are FLOAT, REAL, and DOUBLE PRECISION. Contrast with exact numeric data type.

**association server.** The component of the SQL/MX connectivity service (MXCS) that receives client connection requests for database access and associates the requestor with an MXCS server to execute the requested access. The association server also supplies information about the current state of various MXCS servers. See MXCS and MXCS server.

**atomicity.** A state in which all actions in a transaction happen or none happen. If one part of a transaction fails, the entire transaction fails. In SQL/MX transactions, a state in which the effects of an SQL statement happen or do not happen. When they do not happen, an error is returned to the application and the transaction is not aborted. See also statement atomicity.

**attribute.** A name and value that describe a characteristic of an object, column, data type, file, or MXCI session. See also column attribute, data type attribute, file attribute, heading, and session attribute.

**audited table.** A table flagged for auditing by the HP Transaction Manager Facility (TMF) product. The TMF product monitors all transactions against an audited table in preparation for possible transaction backout or volume recovery.

**authorization ID.** A unique user ID in the NonStop operating system environment. Each user authorized to log on to a node is identified by a user ID that consists of a group and user identification.

**automatic SQL recompilation.** The SQL recompilation in memory of a program or SQL statement invoked by the SQL executor at run time. Automatic recompilation occurs, for example, when an SQL object definition has changed since the program's last explicit SQL compilation.

# B

**backout.** To restore a file, database, or system to its previous condition by removing changes. In transaction processing, to remove all the database changes performed by a transaction that was aborted.

**base table.** A table that has physical existence: that is, a table stored in a file. Contrast with Visual Query Planner. See also file.

**base-table predicate.** A predicate evaluated by the Data Access Manager on rows in the base table.

**binder.** The second phase in the SQL compiler. This portion of the compile process takes the syntactically correct query tree, translates logical names to physical names, and performs semantic checks. The binder also expands views that are referenced in the query, looks up metadata for SQL objects, and produces a semantically correct query tree.

**built-in function.** A function defined by and part of the database that accepts arguments, performs a specific operation, and returns one or more values. For example, the AVG function is a built-in function in SQL/MX that returns the average of a set of numbers.

**bulk move.** The most efficient type of data transfer used by SQL/MP or SQL/MX. The SQL executor selects a bulk move if one or more fields can be moved from a source (for example, a SQL table in DP2 or a buffer in the master executor) to a target (for example, an executor buffer or application host variable) by doing a string copy of all the fields in one operation. The same is true for target-to-source (for example, application host variable to executor buffer) data transfers. A bulk move is selected if the source and target have the same data attributes (length, data type, scale, and so on.) and if all the source fields are aligned the same way as the target. When a bulk move is not selected, each source field is moved or converted to the target one field at a time.

# C

**C/C++ preprocessor.** See SQL preprocessor.

**call-level interface (CLI).** A library of function calls that supports SQL statements.

**CASE expression.** A conditional expression you can use to evaluate a set of conditions and obtain a value based on the condition in the set that is true.

**CAST specification.** A specification or function that converts data to the data type you specify.

**catalog.** A named logical object that contains descriptions of a set of schemas.

**catalog reference.** An indicator that a catalog is known to a node and possibly related to catalog references on other nodes. A catalog reference can be established on a node by creating a catalog with a catalog name that is unique on that node or by registering a catalog that has been created on another node to the node in question.

**channel.** A database table used by applications specifying stream access to subscribe to newly published entries. The entries remain in the table. All subscribers receive new entries.

**character data type.** A data type for data composed of characters, stored as either a fixed-length or variable-length character string. Contrast with numeric data type.

**character expression.** An SQL value expression built from operands such as string literals, column names with character values, character value functions, host variables or any set functions, scalar subqueries, CASE expressions, or CAST specifications that return character values.

**character set.** An attribute, associated with a literal or host variable, that can be one of the following: ISO88591, the default value character set for character data types; KANJI, the double-byte character set widely used on Japanese mainframes; or KSC5601, the double-byte character set required on systems used by government and banking within Korea.

**child node.** A node that receives and performs work on behalf of a distributed transaction. The node that sent the transaction to the child node is known as the “parent node.” During the course of the transaction, the child node may send the transaction to yet another node, becoming then a parent node.

**class.** See Java class.

**CLI.** See call-level interface (CLI) or command-line interface (CLI).

**client.** A software process, hardware device, or combination of the two that requests services from a server. Often, the client is a process residing on a programmable workstation and is the part of an application that provides the user interface. The workstation client might also perform other portions of the application logic. See also server.

**client application.** An application typically running on a client workstation accessing data on a server. A client application is sometimes called a client.

**client/server architecture.** A computer architecture that divides work between a client and a server. The client provides application and user interface resources, and the server stores, retrieves, and protects data.

**clustering key.** A storage key made up of a set of columns in a table whose values act as logical row-ids. The set of columns that make up the clustering key must guarantee uniqueness. See also primary key.

**COBOL preprocessor.** See SQL preprocessor.

**collation.** A convention that governs the ordering of any two character strings formed with characters from their character sets. SQL/MX supports only the default collation that is based on binary ordering. See also character set.

**column.** A vertical component of a table; the relational representation of a field in a record. A column contains one data value for each row of the table. Contrast with row.

**column attribute.** An attribute that describes the data that a column can contain and how the data is stored. Column attributes include data type, default value, heading, row order (ascending or descending), nullable, and UPSHIFT.

**column default value.** The value inserted in a row when an INSERT statement omits a value for a column.

**column heading.** Text specified in a HEADING clause in a CREATE TABLE or ALTER TABLE statement to replace the column name in an MXCI display.

**column name.** An SQL identifier used to identify a column in a table. A table name or correlation name can qualify the column name to make it unique within an SQL statement.

**command file.** See OBEY command file.

**command-line interface (CLI).** An interface between a software subsystem and a user in which the user enters computer-language commands at a prompt. MXCI and SQLCI are examples of command-line interfaces.

**committed transaction.** A transaction that successfully completed its changes to the database.

**comparable data types.** Data types such that a value of one data type can be compared to a value of the other data type. See also compatible data types.

**comparison predicate.** A predicate that compares the values of sequences of expressions, or the values of sequences of row values that are the result of row subqueries.

**compatible data types.** Data types such that a value of one data type can be assigned to a column or parameter of the other data type, and columns of the two data types can be combined using arithmetic operations. Compatible data types are also comparable. See also comparable data types.

**compilation unit.** All the data declarations and units of code specified for processing by one run of a host language compiler.

**compiled module.** See module.

**concurrency.** Access to the same data at the same time by two or more processes.

**concurrent access.** The ability of two or more processes to gain access to the same data at the same time. See also read uncommitted access and read committed access

**configuration client library.** An administrative library for MXCS with a programmatic interface. It connects to MXCS components and passes administrative commands and responses back and forth.

**configuration server.** The component of the SQL/MX connectivity service (MXCS) that manages the MXCS configuration database for the MXCS subsystem The configuration server also supplies information about the current configuration of the MXCS subsystem.

**connection context.** A Java object that associates the execution of each embedded SQL statement (or SQLJ clause) in a Java program with a particular connection to a database. Each SQLJ clause in a Java program is implicitly or explicitly associated with a connection context.

**constraint.** An object that protects the integrity of data in a table by restricting the values in a particular column or set of columns to those that meet specified conditions. SQL/MX supports the constraints NOT NULL, UNIQUE, PRIMARY KEY, CHECK, REFERENCES, and LOGGABLE.

**correlated subquery.** A subquery that references values retrieved by the outer query. A correlated subquery requires repeated evaluation of the subquery for each row satisfying the outer query. Contrast with noncorrelated subquery.

**correlation name.** An SQL identifier you can associate with a table reference in a SELECT statement. You use a correlation name to distinguish a table or view from another table or view referred to in the statement, to distinguish different uses of the same table, or to make the text of the query shorter. A correlation name can be explicit or implicit; the implicit name is the table or view name.

**crash-label state.** A file state in which file label operations set the state of the file to crash-label until the file label operation is committed. If a total system crash occurs or the disk becomes unavailable, the file label state remains set to crash-label until recovery occurs.

**crash-open state.** A file state in which a file is open in the same circumstances as a file in the crash-label state.

**cursor.** A named pointer used by an application program to indicate the current or next row within a set of rows selected by an SQL query.

**cursor name.** An SQL identifier used to identify a cursor.

**cursor position.** The current position of a cursor before, at, or after a row in a set of selected rows.

**cursor stability.** The guarantee that a row at the current cursor position cannot be modified by another program. For SQL/MX to guarantee cursor stability, you must specify SERIALIZABLE access for the SELECT statement that defines the cursor.

**customizer.** See SQLJ customizer.

# D

**DAM parallelism.** Parallel execution in multiple DAM fragment instances, which is characterized by no-wait communication (asynchronous access).

**DAM process.** An instance of the Data Access Manager (DAM). A separate DAM process manages each volume.

**Data Access Manager (DAM).** The component of SQL/MX that provides access to data. A separate DAM process manages each volume.

**data administrator.** The person who specifies what data is needed by a business or organization; the liaison between the business or organization user and the database administrator.

**data consistency.** The condition of a database when related data values agree with one another according to user-defined criteria. Consistency is one aspect of data integrity. See also data integrity.

**Data Control Language (DCL).** See DCL.

**data control statement.** See DCL statement.

**Data Definition Language (DDL).** See DDL.

**data definition statement.** See DDL statement.

**data fork.** See resource fork.

**data independence.** The ability to change the definition of a database without changing the applications that use the database. Contrast with program independence.

**data integrity.** The condition of a database when its data values are accurate, valid, and consistent according to rules established for changing the database. Data consistency is one aspect of data integrity. Contrast with definitional integrity. See also integrity constraint.

**Data Manipulation Language (DML).** See DML.

**data manipulation statement.** See DML statement.

**data source.** The data a user wants to access and the associated operating system, database management system (dbms), and network platform used to access the dbms. To manage ODBC client access to SQL/MX data, you can add, configure, and delete ODBC/MX data sources on both the HP NonStop system and the client computer.

**data type.** A set of valid values with an associated data format. All values in a column must conform to the format of the column's data type. Expressions, functions, host variables, literals, and parameters also have data types. All values passed as parameters to a user-defined routine must conform to the format of the parameter's data type. SQL/MX provides data types for character strings of fixed and varying length, binary and decimal numbers, datetime values, and intervals. See also approximate numeric data type, data type attribute, datetime data type, exact numeric data type, and INTERVAL data type.

**data type attribute.** A format associated with a data type for a column, expression, function, host variable, literal, or parameter.

**database.** A collection of interrelated data with a given structure for storing and providing, on demand, data for multiple users. See also SQL/MX executor.

**database administrator.** See DBA.

**database object.** See SQL object.

**datetime data type.** The DATE, TIME, or TIMESTAMP data type. Datetime data consists of formatted values for one or more of these fields: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND.

**datetime expression.** An SQL value expression built from operands such as datetime or interval literals, column names with datetime or interval values, datetime or interval value functions, host variables or any set functions, scalar subqueries, CASE expressions, or CAST specifications that return datetime or interval values.

**datetime literal.** A literal that has a DATE, TIME, or TIMESTAMP data type. SQL/MX supports a DATETIME literal for insertion into DATETIME columns in an SQL/MX table.

**DBA.** Database administrator. The person who defines or manages a database or controls access to a database.

**DCL.** Data Control Language. The set of data control statements within the SQL language.

**DCL statement.** A statement used to control process resources such as locks and cursors. Contrast with DDL statement and DML statement.

**DDL.** Data Definition Language. The set of data definition statements within the SQL language.

**DDL statement.** A statement used to define, delete, or modify the SQL definition of an object, catalog, or to change the authorization to use the object, catalog, or schema. Contrast with DCL statement and DML statement.

**deadlock.** A block to data access caused by processes contending for the same data. For example, deadlock occurs when process A has locked one row and is waiting for another row already locked by process B, and process B is waiting for the row already locked by process A.

**decision support application.** An application that allows users to query (and possibly update) a database to make business decisions often based on summarization or statistical analysis of transaction data.

**definitional integrity.** The condition of a database when its file labels and metadata tables contain consistent definitions of objects. Contrast with data integrity.

**delimited identifier.** Names consisting of character strings that appear within double quotes ("), consisting of alphanumeric characters and other characters except for the at sign @, forward slash /, backward slash \, circumflex ^. Delimited Identifiers are case-sensitive. See also regular identifier.

**dependency.** The use of an object by another object.

**dependent object.** An object whose definition depends on the definition of a base or underlying object; for example, an index is dependent on a table.

**dequeue.** To read and delete entries with a single operation by using a SELECT statement with an embedded delete. This dequeue operation is sometimes referred to as a destructive select. Alternately, a SELECT statement with an embedded update operation allows an application to dequeue entries without actually deleting the entry from the database table. In this case, the entries are marked for archiving.

**derived column.** An SQL value expression that appears as an item in the select list of a SELECT statement. An explicit name for a derived column is an SQL identifier associated with the derived column.

**derived table.** A query that is nested in the FROM clause of another query.

**directive.** A statement that gives instructions to a compiler.

**dirty reads.** Reads of data as it is modified by a concurrent process. Compare with nonrepeatable reads and phantom reads.

**disk directory.** A directory that contains the file labels for each SQL table, view, and index on a disk volume and the location of each object on the disk.

**Disk Process 2 (DP2).** See Data Access Manager (DAM)

**distributed database.** A database whose objects reside on more than one node in a network of nodes and whose objects can be accessed from any node in the network.

**distributed transaction.** A transaction that accesses data located on multiple nodes of a network, or one that accesses data by means of requesters or servers at multiple nodes of a network.

**distribution service.** When used in SQL/MX error messages, refers to the SQL/MX file system.

**DML.** Data Manipulation Language. The set of data manipulation statements within the SQL language.

**DML statement.** A statement used to select, update, insert, or delete rows in one or more tables. Contrast with DCL statement and DDL statement.

**dynamic input parameter.** A question mark (?) in a dynamic SQL statement that serves as a placeholder for a value substituted when the statement executes within a host language program. An SQL statement that uses dynamic input parameters can be compiled without the actual input values.

**dynamic SQL.** A programming technique that enables an application program to construct, compile, and execute an SQL statement that is unknown until the program is executing. Dynamic SQL supports interactive statements entered at the command line. Contrast with static SQL.

# E

**ELF.** See executable and linking format (ELF).

**embedded module definition.** Contains SQL/MX-specific information for each embedded SQL statement of a program. For an embedded SQL program in C or COBOL, the embedded module definition results by default from language compiling a module definition source and linking it into a self-contained application executable. For an SQLJ program, the embedded module definition results by default from customizing a profile (`.ser` file) and is stored in the extended profile. The embedded module definition is input for the module compiler. For details, see the *SQL/MX Programming Manual for C and COBOL* or the *SQL/MX Programming Manual for Java*. Contrast with module definition file (MDF).

**embedded SQL.** SQL statements in a host language program.

**embedded SQL program.** See SQL program.

**enqueue.** To insert entries into a queue by using an INSERT statement.

**entry-sequenced file.** A file in which each new record is stored at the logical end of the file in chronological sequence. The primary key is a system-generated record address.

Records can be added or updated but not deleted, but variable-length values cannot be shortened or lengthened. Entry-sequenced files can be used only with SQL/MP. Contrast with key-sequenced file.

**entry-sequenced table.** A table stored in an entry-sequenced file and whose primary key is a system-generated record address. Entry-sequenced tables can be used only with SQL/MP tables. Contrast with key-sequenced table.

**environment variable.** A variable that determines the characteristics of an operating system environment for a given user. A user or program can set or change environment variables. Commands or shell scripts evaluate environment variables by substituting the name of the variable with its assigned value.

**ESP.** Executor server process. A process that executes a portion of a parallel execution plan.

**ESP parallelism.** Any parallel plan with at least one ESP plan fragment. ESP parallelism occurs when a plan fragment executes within a special process called the executor server process (ESP).

**exact numeric data type.** A numeric data type that has both precision and scale. The precision is a positive integer that represents the number of significant decimal digits. The scale is a nonnegative integer. The SQL exact numeric types are DECIMAL, INTEGER, LARGEINT, NUMERIC, and SMALLINT. Contrast with approximate numeric data type.

**exception condition.** An abnormal or unexpected result of a programmatic operation.

**exchange node.** Identifies process boundaries for ESP parallelism and DAM parallelism.

**exclusive access.** See repeatable read access or serializable access.

**exclusive lock.** A lock exclusion mode in which only the holder of the lock can access the locked data, except with read uncommitted access. Contrast with shared lock.

**executable.** A prepared application, linked and ready to be loaded from an OSS shell prompt, TACL prompt, or a translated Java class, ready to be interpreted by a Java Virtual Machine (JVM).

**executable and linking format (ELF).** A standard format used for POSIX object files. TNS/R native object files are in ELF format with HP extensions.

**execution characteristics.** The characteristics of an SQL statement's execution plan that have no effect on the statement's semantics. Examples include a plan's performance, node autonomy, resource use, and locking strategy.

**execution context.** A Java object that provides some control over the execution of an embedded SQL statement (or SQLJ clause) in a Java program and allows the retrieval

of information about the execution upon completion. Each SQLJ clause in a Java program is implicitly or explicitly associated with an execution context.

**execution plan.** An execution method, including the semantics and execution characteristics, for a single compiled SQL statement. The SQL compiler stores the execution plan in an SQL module during explicit compilation, and the SQL executor uses the plan to execute the SQL statement at run time. Dynamic SQL statements create an execution plan when the dynamic SQL statement is executed. Compiled programs typically include many plans.

**execution-time name resolution.** The resolution of a name of an SQL object in an SQL statement performed at statement execution time rather than during explicit SQL compilation.

**executor.** The SQL/MX component that executes compiled SQL statements that access database tables, views, and stored procedures. The executor executes a DML statement by using the query execution plan chosen by the optimizer. If necessary, the executor calls the SQL compiler to recompile a program or statement.

**executor predicate.** A predicate that must be evaluated by the executor.

**executor server process (ESP).** See ESP.

**EXPLAIN function result.** A table-valued result of the EXPLAIN function that returns information about query operators describing the access plan for a DML statement. The entries in the table describe an operator tree. See also EXPLAIN operator tree.

**EXPLAIN operator tree.** A tree structure that represents operators used in an access plan as nodes (events in the plan), with at most one parent node for each node in the tree, and with only one root node. Each node might have subordinate nodes.

**explicit correlation name.** A correlation name specified in the FROM clause of a SELECT statement as an alternate name for a table or a view. Contrast with implicit correlation name.

**explicit SQL compilation.** The SQL compilation of a program initiated by running the SQL compiler (mxcmp). The SQL compiler generates a program. Contrast with automatic SQL recompilation.

**expression.** A representation of a value. An SQL value expression has a data type: character, datetime, interval, or numeric.

**extended profile.** `A.ser` file containing the customized profile that the SQLJ customizer generates.

**externally qualified module name.** The three-part module name (`catalog.schema.module`) can contain these externally qualified attributes: table set, version, and group. The module name is externally qualified by using specific preprocessor or customizer options on the command line. For example, the module

named CAT.SCH.MOD, when externally qualified with the table set tabset1, version ver3, and group grp2, becomes CAT.SCH.GRP2^MOD^TABSET1^VER3. For more information, see the *SQL/MX Programming Manual for C and COBOL* or the *SQL/MX Programming Manual for Java*.

# F

**field.** A portion of a record reserved for a specific item of information. In relational terms, each column of a table represents one field. In a datetime or INTERVAL column value, a datetime field or INTERVAL field, such as HOUR, is a specific part of the column.

**file.** The physical storage for a table, index, table partition, or index partition. The creation of a table or index implicitly creates a file or files to contain the data, one file for each partition. The names of the storage files are system-generated.

**file attribute.** An attribute that describes a physical characteristic of a file (containing tables, indexes, or partitions) such as block size, record length, the number and sizes of extents, or whether the file is audited. The file attributes potentially affect the performance of applications that use the files.

**file label.** A label, containing file information in encoded form, associated with each table, index, view, table partition, and index partition.

**file organization.** The physical file organization available for tables. See also entry-sequenced file and key-sequenced file.

**first key.** The clustering-key value specified for the first row in a partition. Whether the value in each key column is high or low depends on the ascending or descending collating sequence defined for the column. The value is the lowest in the partition for ascending columns and the highest for descending columns.

**foreign key.** An integrity constraint that requires each value in a column or set of columns to match a value in a related table's UNIQUE or PRIMARY KEY. FOREIGN KEY integrity constraints also define referential integrity actions that dictate what SQL/MX should do with dependent data if the data it references is altered. See also primary key and integrity constraint.

**function.** A routine that returns a value. Contrast with procedure. See also built-in function and routine.

# G

**generator.** The last phase of the SQL compiler. This portion of the compile process takes the optimal query plan and generates the executable query tree for the run-time component.

**grouped select.** A query that generates a single row in the result table by processing a set of rows from the table produced by the FROM clause. The grouping is determined by a GROUP BY clause, a set (or aggregate) function in the select list, or a grouped view in the FROM clause. The SELECT operation can result in one group (if no GROUP BY clause is specified) or a set of groups.

**grouped view.** A view defined with a query that contains a GROUP BY or HAVING clause that is not a subquery, contains a set (or aggregate) function in the select list, or refers to another grouped view in the FROM clause.

**grouping column.** A column specified in a GROUP BY clause.

# H

**hash join.** A join algorithm that creates a hash table for the inner table and joins the outer table by hashing each outer row and looking for matches in the hash table. In some situations, this algorithm performs better than either the sort merge or nested join algorithms.

**hash partitioning.** A method of distributing records among partitions based on a hash function that results in partitions of approximately equal size even if you do not know the range values. With hash partitioning, SQL/MX uses a hash function on the values of the partitioning key, and each record is assigned to a partition based on the result. Records are evenly distributed across the partitions. Compare with range partitioning.

**heading.** A column attribute that specifies one or more lines of text to appear at the top of a column in MXCI query results.

**histogram.** A representation of a relationship in which each value of some dependent variable corresponds to a range of values of the associated independent variable or variables. SQL/MX provides a method for generating histograms that show how data is distributed with respect to a column or a group of columns within a table. See also histogram_intervals

**histogram caching.** The ability of the SQL/MX compiler, when enabled, to cache in memory histograms for tables used in a particular query and reuse them later for statements during the same session.

**histogram_intervals.** User metadata tables that contain logical (table and column level) histogram statistics. The UPDATE STATISTICS statement is used to specify columns or groups of columns within the source table, to generate logical statistics that are stored in the HISTOGRAM and HISTINTS tables. The data in these tables can be accessed by using the SELECT statement. See also histogram.

**host identifier.** A name used in a host program to identify a program element such as a data item, paragraph, or section.

**host language.** Any programming language whose statements can be combined with SQL statements in the same source program. SQL/MX supports C, C++, COBOL, and Java as host languages.

**host language compiler.** The C, C++, COBOL or Java compiler.

**host program.** A source program that contains both host language statements and embedded SQL statements.

**host variable.** A data item declared in the host language program that can be used by both host language statements and embedded SQL statements in the same source program. A host variable provides communication between these two types of statements. An input host variable transfers data from a program to a database, and an output host variable transfers data from a database to a program. A host variable has a corresponding SQL data type.

**hybrid hash join.** A hybrid hash join joins the data from its children. It creates a hash table for the inner child and joins the outer table by hashing each outer row and looking for matches in the hash table. This operator can overflow to disk when the inner table is too large to fit in memory. Equijoins and cross-products are supported by hybrid hash join.

# I

**implicit correlation name.** A table or view name for which no correlation name is specified in the FROM clause of a SELECT statement. Contrast with explicit correlation name.

**IN parameter mode.** The mode of a parameter that passes data to a user-defined routine (UDR). In an SQLJ program, this mode specifies a host variable or expression that passes data to an SQLJ clause. It is the default mode if the host expression is not part of an INTO list or an assignment expression. Contrast with INOUT parameter mode and OUT parameter mode.

**independent parallelism.** An inherent feature of SQL/MX where two or more operators can execute simultaneously and independently. Also called operator parallelism.

**index.** An alternate access path (alternate key) to a table that differs from the primary access path (clustering key) defined for the table at creation time. An index, stored in a key-sequenced file, includes columns for the clustering key and the alternate key.

**index predicate.** A predicate evaluated on rows in an index by the Data Access Manager.

**indicator variable.** An exact numeric data item associated with a host variable and declared in an SQL declare section. A C, C++ or COBOL program uses an indicator variable to insert null into a column, to pass a null value into or out of a user-defined routine, or to indicate whether a target host variable (of any data type) contains null or whether the target host variable (of character data type) is smaller than the source character string.

**inner query.** A subquery nested within a statement or another subquery. See also outer query and subquery.

**inner table.** In the evaluation of a join, the table that the executor accesses after the outer table.

**INOUT parameter mode.** The mode of a parameter that passes data to and accepts data from a user-defined routine. Contrast with IN parameter mode and OUT parameter mode.

**Insert/update/delete.** A group of three operations that are used to change data in SQL/MX tables.

**integrity constraint.** A condition that must be met before rows can be inserted or columns updated in the table for which the constraint is defined. This declarative condition helps to maintain data integrity. See also constraint and data integrity.

**INTERVAL data type.** A data type for year-month and day-time values that represents intervals of time. INTERVAL data consists of formatted values for one or more fields: either YEAR, MONTH, and DAY or HOUR, MINUTE, and SECOND.

**INTERVAL expression.** An SQL value expression built from operands such as interval literals, column names with datetime or interval values, datetime or interval value functions, host variables or any set functions, scalar subqueries, CASE expressions, or CAST specifications that return interval values.

**INTERVAL literal.** A literal that has the INTERVAL data type.

**INTERVAL qualifier.** See range of INTERVAL fields.

**isolation level.** A transaction level of data concurrency that is either read uncommitted, read committed, repeatable read, or serializable. You can set the isolation level of a transaction explicitly by using a SET TRANSACTION statement, by including an access clause in SQL statements, or by using defaults.

**iterator.** A Java object that retrieves rows from the result table of a query in an SQLJ program. An iterator is similar to a cursor, except that an iterator can be passed as a parameter to a Java method. See also named iterator and positioned iterator.

**IUD.** Insert/update/delete. This abbreviation is often used to group together these three operations that change data in SQL/MX tables. See also Insert/update/delete.

# J

**Java class.** A building block of a Java program that consists of a class declaration, variable (or field) declarations, and methods. A Java class serves as a template for Java objects.

**Java Database Connectivity (JDBC).** The Java standard for access to relational databases such as SQL/MP or SQL/MX.

**Java method.** A function defined in a Java class.

**Java object.** An instantiation of a Java class.

**Java Virtual Machine (JVM).** Software that loads, links, verifies, and interprets Java bytecode.

**join.** A database operation that combines two or more tables into a single logical table so that data can be selected from all the tables at once. Types of joins include natural inner (or natural), natural left, natural right, join on, left join on, right join on, merge join, left merge join, soft merge join, nested join, left nested join, hash join, hybrid hash join, left hybrid hash join, and simple hash join.

**join on.** A type of join that joins only rows that satisfy the search condition in the ON clause.

**join predicate.** A predicate that identifies and compares columns in a join operation.

# K

**key predicate.** A predicate that specifies begin-key conditions, end-key conditions, or both to narrow the range of searching an SQL table. Such conditions reduce the number of rows fetched; this reduction of physical I/O operations improves performance.

**key prefix.** A leading (leftmost) contiguous set of columns in the key.

**key-sequenced file.** A file in which each new record is stored in sequence by a primary or clustering key. See also primary key and clustering key. Contrast with entry-sequenced file.

**key-sequenced table.** A table stored in a key-sequenced file, where the storage key can be defined by the user, by both the user and the system (SYSKEY), or by the SYSKEY alone. You cannot update columns in a user-defined clustering or primary key. Columns can be added to the table, rows can be updated or deleted, and variable-length values can be shortened or lengthened. Contrast with entry-sequenced table.

# L

**label.** See file label.

**language manager.** See SQL/MX language manager.

**late name resolution.** A comparison made by SQL/MX to determine whether a new table name specified at run time by a host variable or a DEFINE is the same as a previous table name. When a table name changes in a query, SQL/MX can recompile the query or use its previous access plan depending on the outcome of a similarity check.

**leaf operator.** An operator in an operator tree that has no children. An example of a leaf operator in an operator tree is the scan node. The EXPLAIN row for a relational leaf operator will have NULL value for the left and the right child column.

**left hybrid hash join.** A left hybrid hash join returns an unmatched outer row even when it does not find a match in the inner table. Null values are supplied for the missing inner rows. This operation differs from a hybrid hash join only when it does not find a match in the inner table. See also hybrid hash join.

**left join on.** A type of join that joins rows that satisfy the search condition in the ON clause of a join operation, plus rows from the left table that do not satisfy the search condition.

**left merge join.** A portion of an execution plan that involves a merge join. This operation differs from a merge join only when it does not find a match in the inner table. When no match is found, the left row is still returned, and the data from the right table is set to null. See also sort merge join.

**left nested join.** A portion of an execution plan that involves a nested join. This operation sends each outer (left) row to the inner (right) child. The right child finds all the matches for a row and returns all the matches. If an outer row finds no matches in the inner table, the outer row is returned and nulls are supplied for inner table values. See also nested join.

**literal.** A constant in an expression or statement. An SQL literal can be a datetime, INTERVAL, numeric, or character string literal.

**live upgrade.** The ability to replace the program file of a running process with an identically named file. The HP NonStop operating system supports live upgrades for applications that can be loaded directly from the HP NonStop Open System Services (OSS) or NonStop operating system command line, but not Java applications, which are interpreted by a Java Virtual Machine (JVM).

**local autonomy.** The principle that users of a node in a network can perform operations on objects on that node even when other nodes are not available and also manage nodes independently.

**local transaction.** A transaction that is processed by a single transaction manager on a single node. Contrast with distributed transaction.

**location-independent ANSI name.** An SQL/MX catalog, schema, or database object name that is subject to visibility rules. ANSI name lookup resolves a name starting with the system schema tables on the local node. The name does not include an expand node name and cannot be guaranteed to unambiguously identify a database object in an Expand network.

**lock.** A mechanism that coordinates concurrent access to the same data. See also exclusive lock, lock duration, lock granularity, and shared lock.

**lock duration.** The period of time during which a lock is held.

**lock escalation.** The replacement of all the row locks on a partition or table by a single partition lock or table lock when the number of row locks approaches a limit. See also lock granularity.

**lock exclusion mode.** A lock attribute that determines whether any process except the lock holder can access the locked data. See also exclusive lock and shared lock.

**lock granularity.** The size of a lockable unit. A lockable unit can be a single row (row lock), a subset of the rows in a table, a partition of a table or a nonpartitioned table (partition lock), or an entire partitioned table (table lock).

**lock holder.** The process that acquires an exclusive or shared lock on data, sometimes called lock owner.

**lockable unit.** The amount of data that can be protected by a single lock. Tables, partitions of tables, subsets of rows in a table, and single rows are lockable units. See also lock granularity.

# M

**MACL.** SQL/MX administrative command library. A library of administrative commands, linked into MXCI, that supports manipulation of logical objects defining the user-visible attributes of the SQL/MX connectivity services (MXCS). See MXCS.

**MDAM.** MultiDimensional Access Method**.** An optimized scan method that provides improved performance for queries that contain range or missing predicates on key columns.

**MDF.** See module definition file (MDF).

**merge join.** See sort merge join.

**metadata tables.** Tables that store information about all SQL objects within a database. The system metadata is created and maintained by the system as users create, alter, or drop objects. Contrast with user metadata tables.

**method.** See Java method.

**MGSS.** See Module Group Specification String (MGSS).

**Microsoft ODBC.** Open Database Connectivity. A specification for an application programming interface (API) that defines a standard set of routines that an application can use to access a database. ODBC is based on the call level interface (CLI) specifications from X/Open and ISO/IEC for database APIs and uses structured query language (SQL) as its database access language. By calling ODBC functions through database-specific drivers, which are loaded at run time, an application can access different databases by using the same source code. See also NonStop ODBC/MX association server.

**Microsoft ODBC driver manager.** An ODBC component that manages access to ODBC drivers for ODBC applications. The driver manager loads and unloads ODBC drivers and passes calls for ODBC functions to the correct driver. ODBC/MX uses the Microsoft 3.0 (or later) Driver Manager.

**Microsoft ODBC server.** An ODBC component in the client/server architecture that stores, retrieves, and protects data in the database. An ODBC/MX server implements core ODBC function calls by using the SQL/MX call level interface (CLI) to access an SQL/MP database.

**mixed-workload environment.** A computing environment characterized by sometimes conflicting demands for I/O and processor resources. NonStop SQL/MX is designed to overcome issues associated with mixed workloads executing concurrently on the same server. Priorities can be assigned to individual workloads (queries, transactions, and management commands) based on the urgency of the information or the expected execution cost in terms of I/O and processor cycles. The priority of an individual workload dictates the processor cycles and I/O allocated to it relative to other workloads executing at the same time on the same server. This priority is carried with each resource request made on behalf of the query or application, even when requests are passed to other processors.

**module.** Contains the compiled and optimized execution plans of SQL statements from an embedded SQL program. The module is necessary for the execution of statically compiled programs.

**module compiler.** A utility that takes a self-contained (or partly self-contained) application executable and SQL/MX compiles its embedded module definition.

**module definition file (MDF).** Contains SQL/MX-specific information for each embedded SQL statement of a program. For embedded SQL in C or COBOL, the MDF is the output of the SQL preprocessor when you specify certain options on the command line. For embedded SQL in Java (SQLJ), the MDF is the output of the SQLJ customizer when you specify a certain option on the command line. The MDF is the input file for the SQL/MX compiler. For details, see the *SQL/MX Programming Manual for C and COBO*L or the *SQL/MX Programming Manual for Java*. Contrast with embedded module definition.

**module definition source.** The module definitions emitted by the preprocessor as extra source code in the annotated output source file.

**MODULE directive.** An embedded SQL statement that specifies the name of a module.

**module file.** The file in which a module's compiled SQL plans are stored.

**Module Group Specification String (MGSS).** A regular identifier used by the Module Management subsystem to represent a group. The C/C++, and COBOL preprocessors and SQLJ Customizer append a leading circumflex (^) character and the MGSS to a processed module name. See also processed module name.

**module language.** A language for expressing SQL operations in pure SQL syntactic form used in module definition files and embedded module definitions.

**module management qualified module name.** See externally qualified module name.

**Module TableSet Specification String (MTSS).** A regular identifier used by the Module Management subsystem to represent a table set or target. The C, C++, and COBOL preprocessors and SQLJ Customizer append a leading circumflex (^) character and the MTSS to a processed module name. See also processed module name.

**Module Version Specification String (MVSS).** A regular identifier used by the Module Management subsystem to represent a version. The C, C++, and COBOL preprocessors and SQLJ Customizer append a leading circumflex (^) character and the MVSS to a processed module name. See also processed module name.

**MPALIAS table.** A SQL/MP user metadata table created by SQL/MX that contains mappings from logical object names to physical NonStop operating system locations. The MPALIAS table is obsolete in Release 2.0. Starting with Release 2.0, these mappings are created in existing catalogs and schemas and are stored in SQL/MX metadata as SQL/MX aliases that consist of the catalog, schema, and name. See also alias mapping and user metadata tables.

**MTSS.** See Module TableSet Specification String (MTSS).

**MultiDimensional Access Method (MDAM).** See MDAM.

**MVSS.** See Module Version Specification String (MVSS).

**MXCI.** SQL/MX conversational interface. MXCI enables a user to execute SQL/MX statements and commands, and operate database utilities without embedding the SQL statements in an application. Contrast with programmatic SQL.

**MXCS.** SQL/MX connectivity services. The services provided by the NonStop operating system that enable connectivity APIs, such as ODBC and JDBC, to access SQL/MX.

**MXCS server.** The component of the SQL/MX connectivity service (MXCS) that connects to a remote SQL/MX client.

**MXCI parameter.** A name (up to 128 alphanumeric characters and underscore) preceded by a question mark (?) in an SQL statement (to be run within an MXCI session) that serves as a placeholder for a value substituted when the statement executes. See also SQL parameter.

**MXCI session.** See session.

**mxcmp.** The SQL compiler. See SQL compiler.

**MXSQLC.** The SQL/MX C/C++ preprocessor. See SQL preprocessor.

**MXSQLCO.** The SQL COBOL preprocessor. See SQL preprocessor.

**MXUDR.** The SQL/MX UDR server. See also SQL/MX UDR server.

**MXV.** See SQL/MX software version (MXV).

# N

**name resolution.** The process of resolving names in a statement by mapping logical names to their physical names and expanding partially qualified object and schema names by using the current default values.

**namespace.** A logical naming scheme for grouping related types. SQL/MX supports the following namespaces for externally accessible objects: table, index, constraint, trigger, and module.

**named iterator.** A type of iterator where the names for each iterator column are specified in the iterator declaration and must match the names of the corresponding columns in the result table. Contrast with positioned iterator. See also iterator.

**national character set.** A special character set defined during the SYSGEN of an operating system or by the default attribute NATIONAL_CHARSET. This character set governs the interpretation of the character string literal with the N prefix and the NCHAR keyword used in a CAST operation. See also character set.

**natural join.** A type of join that joins rows only where the values of all columns that have the same name in both tables match. Also called natural inner join.

**natural left join.** A type of join that joins rows where the values of all columns that have the same name in both tables match, plus rows from the left table that do not meet this condition.

**natural right join.** A type of join that joins rows where the values of all columns that have the same name in both tables match, plus rows from the right table that do not meet this condition.

**nested join.** A join method that compares each row from the outer table with each row from the inner table to find a match between the specified join columns from the two tables.

**node.** A uniquely identified computer connected to one or more other computer systems in a network.

**noncorrelated subquery.** A subquery that does not reference or depend on the result of the outer query. A noncorrelated subquery can be evaluated once and the result used repeatedly for the outer query. Contrast with correlated subquery.

**nonrepeatable reads.** Reads of committed values for the same item at different times or of items that no longer exist. Compare with dirty reads and phantom reads.

**nonsensitive command.** A SQL/MX connectivity service (MXCS) administrative command that returns only the status or attributes of objects in the MXCS subsystem being administered. Contrast with sensitive command.

**NonStop Data Access Manager.** See Data Access Manager (DAM).

**NonStop ODBC driver.** A Microsoft ODBC component needed for an ODBC application to communicate with a server. ODBC drivers are DLLs and are loaded on demand.

**NonStop ODBC/MX.** A NonStop software product that allows client applications developed for the ODBC application programming interface (API) to access an SQL/MX database in an HP NonStop system running SQL/MX.

**NonStop ODBC/MX association server.** The ODBC/MX component that starts and manages the ODBC/MX servers in HP NonStop systems and associates an ODBC client with a specific ODBC/MX server.

**NonStop ODBC/MX automatic startup option.** An ODBC/MX startup option that causes an ODBC/MX data source to be started automatically when the ODBC/MX service is started. See also NonStop ODBC/MX manual startup option.

**NonStop ODBC/MX configuration server.** The ODBC/MX component that manages the ODBC/MX configuration data in the system catalog schema.

**NonStop ODBC/MX default data source.** A data source named TDM_Default_DataSource that ODBC/MX automatically creates in an HP NonStop system. ODBC clients can connect to TDM_Default_DataSource and use ODBC/MX without having to add and configure an ODBC/MX server data source on the NonStop system.

**NonStop ODBC/MX driver.** The ODBC/MX component that implements ODBC function calls to enable an ODBC client application to access an SQL/MP database through an SQL/MX server.

**NonStop ODBC/MX governing action.** The action taken by ODBC/MX when the limit is exceeded for a resource management policy attribute. Examples of governing actions are COMMIT and ROLLBACK.

**NonStop ODBC/MX governing attribute.** A resource management policy entity that is used to compare an aspect of an SQL/MX query to a limit value. An example of a governing attribute is the estimated cost of a query.

**NonStop ODBC/MX manual startup option.** An ODBC/MX startup option that requires the user to manually start an ODBC/MX data source. See also NonStop ODBC/MX automatic startup option.

**NonStop ODBC/MX resource management policy.** An attribute-value pair that associates a governing attribute name with a test limit value and a governing action that ODBC/MX takes if the limit value is exceeded.

**NonStop SQL/MP.** See SQL/MP.

**NonStop SQL/MX.** See SQL/MX.

**NonStop SQL/MP conversational interface.** See SQLCI.

**NonStop SQL/MX conversational interface.** See MXCI.

**NonStop TMF product.** See Transaction Management Facility (TMF).

**normalizer.** A portion of the SQL compiler that takes the semantically correct query tree and performs certain unconditional transformations, such as constant folding, subquery elimination, and recognizes equivalent expressions by representing equivalent groups of values. These transformations make the query representation suitable for optimization. The transformations are not based on cost. Produces the normalized query tree as input to the optimizer (semantically correct query tree in canonical form).

**NSM/web.** A Web-based tool for managing SQL/MX connectivity services (MXCS) and database objects.

**null.** A missing or unknown SQL value.

**numeric data type.** A data type for data composed of numbers, stored either in binary form or as ASCII characters. Contrast with character data type.

**numeric expression.** An SQL value expression built from operands such as numeric literals, column names with numeric values, numeric value functions, host variables or any set functions, scalar subqueries, CASE expressions, or CAST specifications that return numeric values.

**numeric literal.** Data composed of numbers. A numeric literal can be exact or approximate.

# O

**OBEY command file.** A text file containing statements or commands you can execute by using an OBEY command.

**object.** See Java object or SQL object.

**object feature version (OFV).** The description of features actually used by a database object. The version can change as features are added or removed from an object during DDL or utility operations.

**object name.** An object identifier that is either a three-part logical name or a four-part physical NonStop operating system name. A logical name has the form *catalog.schema.name*, where each of the three parts is an SQL identifier, and a NonStop operating system name has the form [*node*.][[*$volume*.]*sub-vol.*]*filename*. See also alias mapping, catalog, and schema.

**object schema version (OSV).** The schema version of a database object's schema. The version is assigned to the object at create time and changes when the object's schema is upgraded or downgraded.

**OFV.** See object feature version (OFV).

**OLT optimization.** A method of optimizing online transaction processing performance and influencing SQL query performance. Some strategies involve simple changes to system defaults, and others are more complex and involve investigating the query plan and making appropriate changes to the query or query plan.

**online transaction processing (OLTP).** A method of processing transactions in which entered transactions are immediately applied to the database. The information within the database is readily available to all users through online screens and printed reports. The transactions are processed while the requester waits, as opposed to queued or batched transactions, which are processed at a later time.

Online transaction processing can be used for many different kinds of business tasks such as order processing, inventory control, accounting functions, and banking operations.

**online transaction processing (OLTP) application.** An application in which many users can update data simultaneously, recording the changes in the database as they are entered. OLTP applications typically use simple queries that require quick response time against tables containing a mix of read and write requests.

**Open Database Connectivity (ODBC).** See Microsoft ODBC and NonStop ODBC/MX association server.

**operator.** Represents an event in a query plan. Also called a node.

**operator tree.** A structure that represents operators used in a query plan as nodes, with at most one parent node for each node in the tree, and with only one root node.

**optimization.** See query optimization.

**optimizer.** A component of the SQL compiler that chooses query execution plans for retrieving data from a database.

**ordered stream.** An application specifying stream access can retrieve queue entries in the order of a user-defined clustering key or a system-generated clustering key, the SYSKEY, if there is no user-defined clustering key. An application can also retrieve entries in the order of a column value if a secondary index has been created for that column. Because streams do not return an end-of-data condition, normal sort operations cannot be used to sort the retrieved entries. Instead, the sort order must be materialized by the clustering key or a secondary index.

**OSV.** See object schema version (OSV).

**OUT parameter mode.** The mode of a parameter that accepts data from a user-defined routine (UDR). In an SQLJ program, this mode specifies a host variable or expression that accepts data from an SQLJ clause. It is the default mode if the host expression is part of an INTO list or an assignment expression. Contrast with IN parameter mode and INOUT parameter mode.

**outer query.** A statement containing a subquery or a subquery containing another subquery. See also inner query and subquery.

**outer reference.** A reference from an inner query to a table or view defined in an outer query. See also inner query and outer query.

**outer table.** In the evaluation of a join, the table that the executor accesses first.

# P

**parallel execution.** The ability of SQL/MX to execute large queries in parallel, using multiple CPUs simultaneously.

**parallel sort.** A sorting process that shortens the elapsed time for sorting large files by distributing the workload to multiple processors and by using scratch files on multiple disks.

**parameter.** A placeholder for an argument, which is a value or reference passed to a routine, command, or program by the caller. See also SQL parameter.

**parent node.** A node that sends a distributed transaction to another node to perform work on it. Contrast with child node.

**parser.** Performs syntax checks and translates the SQL/MX query into a syntactically correct query tree.

**partition lock.** A lock held by a process on all the rows in a partition of a table or in a nonpartitioned table. Contrast with row lock and table lock.

**partitioned object.** A table or index that resides on more than one disk volume, possibly on more than one node.

**partitioned parallelism.** A feature of SQL/MX that provides the ability to divide the data to be processed into partitions and work on each partition in parallel.

**phantom reads.** Reads of different sets of committed values that satisfy the same predicate at different times. Compare with dirty reads and nonrepeatable reads.

**physical file attribute.** See file attribute.

**pipelined parallelism.** An inherent feature of SQL/MX that interconnects all operators by pipes, with the output of one operator being piped as input to the next operator.

**plan fragment.** A portion of a query execution plan that executes within a single process such as DAM, ESP, or master executor.

**policy file.** A file that instructs the Java security manager to allow certain restricted operations.

**positioned iterator.** A type of iterator where the data types of columns in the iterator declaration correspond in order to the data types of columns in the result table. Contrast with named iterator. See also iterator.

**precision.** A positive integer that represents the number of significant digits, the maximum number of digits in the evaluation of an arithmetic expression. Contrast with scale.

**predicate.** A Boolean expression that determines an answer to a question about a value or group of values. A predicate returns true, false, or, if the question cannot be answered, unknown. You use predicates within search conditions to choose rows from tables or views.

**preprocessor.** See SQL preprocessor.

**primary key.** A column or set of columns that define the uniqueness constraint for a table. The columns cannot contain nulls, and there is only one primary key constraint on a table. See also clustering key.

**primary partition.** The partition of a partitioned table or index that contains the lowest set of key values if the collating sequence is ascending, or the highest set of key values if the collating sequence is descending.

**procedure.** A routine that does not return a value. Contrast with function. See also routine.

**process.** An executing object program, including code and data spaces.

**process name.** An operating system name that can be assigned to a process when the process is created. A process name uniquely identifies a process or process pair on a node.

**processed module name.** In the context of module management, the module name that results after it has been qualified with the default catalog and schema (if none was given in any MODULE directive), after it has been synthesized based on a timestamp (if no MODULE directive was given). In product versions prior to SQL/MX 1.8, the delimited identifier characters can be transformed. In SQL/MX 1.8, the delimited identifier characters, except for the beginning, ending, and any embedded quotes, are preserved as is.

**profile.** A `.ser` file that contains serialized Java object code, including SQLJ clauses and host variable attributes. The SQLJ translator generates one profile for each connection context in the SQLJ source file (`.sqlj` file).

**program independence.** The ability to change an application without changing the database. Contrast with data independence.

**programmatic SQL.** The SQL statements and directives that can be embedded in a host-language program; the application programming interface for SQL/MX. Contrast with MXCI.

**publish.** To insert rows of data into a queue or channel by using an INSERT or UPDATE statement.

# Q

**query.** Generally, a request for data from the database; specifically, the execution of a SELECT statement, which requests columns and rows from one or more tables and views. In the context of the SQL/MX optimizer, a query is a request for data access through any DML statement.

**query execution plan.** See access plan and execution plan.

**query expression.** A syntactical element that defines a derived table. A derived table can be used as a table reference in the FROM clause of a SELECT statement. See also table reference.

**query optimization.** The selection by the optimizer of the best available query execution plan for each DML statement, including the access path, join order, type of join, sort strategy, and so forth.

**query plan caching.** A feature of the SQL/MX compiler that provides the ability to cache the plans of certain queries.

**queue.** A database table used by applications specifying stream access to dequeue newly published entries. The entries are removed from the table. Only a single subscriber receives a new entry.

# R

**range partitioning.** A method of distributing records among partitions based on ranges of key values. With range partitioning, you define key ranges for each partition, and each record is assigned to the partition whose range includes the value of its partitioning key. Compare with hash partitioning.

**range of INTERVAL fields.** The subset of fields, called an INTERVAL qualifier, that compose a value of type INTERVAL. The range can include either year-month or day-time fields, but not both. Some sample ranges are YEAR, DAY TO HOUR, and MINUTE TO SECOND.

**RDF.** See Remote Duplicate Database Facility (RDF).

**read committed access.** An access option for any DML statement that requires that the data being accessed is from committed rows. SQL/MX requires that a lock can be acquired on the data requested by the DML statement, but does not actually lock the data, thereby reducing lock request conflicts. If a lock cannot be granted (implying that the row contains uncommitted data), the DML statement request waits until the lock in place is released. Contrast with read uncommitted access and serializable access.

**read uncommitted access.** An access option for the SELECT statement that gives users immediate access to data when a transaction in progress might be changing the data. The data accessed need not be from committed rows. With read uncommitted access, a process does not acquire a lock on the data it accesses and does not test for existing locks before reading data. Contrast with read committed access and serializable access.

**recompilation.** See automatic SQL recompilation or explicit SQL compilation.

**record.** A unit of data that can be read from or written to a file; a record is represented as a row in a relational table.

**record locks.** A lock held by a process on one specific record; corresponds to row lock.

**recovery.** The returning of a database file or files to a consistent state.

**referential integrity.** A SQL-92 feature that prevents users or applications from entering inconsistent data. The SQL/MX ALTER TABLE statement allows you to set referential constraints for tables to declare that a column or set of columns (called a foreign key) can contain only values that match those in a column or set of columns specified in the REFERENCES clause. The two columns or sets of columns must have the same characteristics (data type, length, scale, precision). There must also be a UNIQUE or PRIMARY KEY constraint on the column or set of columns specified in the REFERENCES clause. The foreign key is the column or set of columns specified in the FOREIGN KEY clause, immediately following the FOREIGN KEY keywords.

**register.** Establishing a catalog reference on a remote node for a catalog that is visible from the local node.

**regular identifier.** Names beginning with a letter (A through Z or a through z), that can also contain other letters, digits (0-9), or the underbar character (_). Regular Identifiers are not case-sensitive. See also delimited identifier.

**related catalogs.** The relationship between catalogs when objects from one catalog depend upon objects from another catalog because of a trigger, view, or constraint. Related catalogs must have identical visibility; that is, be registered on the same set of nodes.

**relational database.** A database in which data is represented as tables.

**Remote Duplicate Database Facility (RDF).** An HP NonStop data replication product that monitors changes made to a production database on a local (primary) system and maintains a copy of that database on one or more remote (backup) systems.

**repeatable read.** The level of consistency that repeatable read access provides. Contrast with cursor stability. See also repeatable read access.

**repeatable read access.** An access option for transaction consistency that ensures that the data accessed by a transaction cannot be changed by other users until the transaction ends. With repeatable read access, a process tests for existing locks on data before acquiring its own lock and holds the lock until the transaction completes. The implementation for repeatable read access is equivalent to serializable access. Contrast with read uncommitted access and read committed access.

**reserved word.** A word reserved for use by SQL/MX. A reserved word cannot be used as a regular identifier.

**resource fork.** One of two physical files instantiated when an SQL/MX object is created. One file is the data fork, where user data resides, and is similar to an SQL/MP format object. The other, the resource fork, is a new file that contains more structural descriptions of the table. In contrast, SQL/MP keeps all its structure information in its metadata and DAM labels. Because DAM has a limitation on how much label information can be stored, SQL/MX keeps all its run-time metadata information, such as the partition array, in the resource fork. Resource forks are created for all persistent SQL/MX objects, such as tables, SQL/MX temp tables, SQL/MX worktables, and indexes.

**result table.** The output that results from the execution of a SELECT statement.

**right join on.** A type of join that joins rows that satisfy the search condition in the ON clause of a join operation, plus rows from the right table that do not satisfy the search condition.

**routine.** A set of program statements that performs a specific operation. A routine can accept arguments and return values. Functions and procedures are types of routines. See also function, procedure, and user-defined routine (UDR).

**row.** A horizontal component of a table; the relational representation of a record. A row contains one data value for each column in the table. Contrast with column.

**row lock.** A lock held by a process on one specific row. Contrast with partition lock and table lock.

**row value constructor.** A series of expressions enclosed in parentheses or a row subquery (subquery returning a single row). Row value constructors are used as operands of predicates.

**rowset.** A group of rows retrieved by an SQL cursor in a single FETCH operation.

**run unit.** A set of one or more object programs in the same object program file.

# S

**savepoint.** A marker that allows an application to roll back part of a transaction if a minor error is encountered. The application must still commit or roll back the full transaction when it is complete.

**scale.** A positive integer that represents the number of digits to the right of the decimal point in the evaluation of an arithmetic expression. Contrast with precision.

**schema.** A named collection of SQL/MX database objects. Each SQL/MX database object is described in only one schema, although an object can refer to objects described in other schemas. A schema cannot contain other schemas. Each schema is described in a catalog. The schema is the unit of ownership, all database objects in a schema are owned by the schema owner.

**search condition.** A predicate or compound predicate that defines the criteria for selecting data or for including data in a table or view.

**select list.** A list that specifies columns or expressions to include in the result of a query or the columns in which to insert or update data.

**self-contained application.** An embedded SQL C/C++/COBOL application whose sources have been preprocessed by preprocessors without using the -x option. The application is self-contained because it does not need separate module definition files. Its module definitions are in the application's object file.

**self-referencing foreign key constraint.** The referential constraint where a foreign key is referencing the primary key of the same table.

**sensitive command.** A SQL/MX connectivity service (MXCS) administrative command that changes data or objects in the MXCS subsystem being administrated. Contrast with nonsensitive command.

**serializable access.** An access option for any DML statement that requires that the statement and any concurrent process (accessing the same data) execute as if the statement and the other process had run serially rather than concurrently. This option locks all data accessed through the DML statement and holds the locks on data in audited tables until the end of any containing transaction. Serializable access provides the repeatable read capability. Contrast with read uncommitted access and read committed access.

**server.** A process or program that provides services to a client or a requester. A server process is a running instance of a server program. A combination of hardware and software designed to provide services in response to requests received from clients across a network. See also node.

**session.** A particular execution of MXCI from the time MXCI is invoked to the time it is terminated.

**session attribute.** One of a set of attributes consisting of parameters and options whose values describe the current MXCI session.

**set function.** A function that operates on a group or groups of rows retrieved by the SELECT statement or the subquery in which the set function appears. Also called an aggregate function.

**shared lock.** A lock exclusion mode that allows any number of processes to read the same data but prevents any process from writing to the locked unit or reading it with intent to rewrite. Contrast with exclusive lock.

**similarity check.** A comparison made by SQL/MX to determine whether two tables (the previous compile-time table and the new run-time table) are sufficiently similar so that an access plan compiled for the previous table is also an operable plan for the new table. A successful similarity check preserves the original access plan and avoids recompilation.

**simple hash join.** A hash join method in which the inner table is read into a memory-resident hash table. The order of the outer table is preserved. See also hybrid hash join.

**skip conflict access.** Access mode used by Publish/Subscribe that allows transactions to skip rows locked in a conflicting mode by another transaction.

**SMD.** See system metadata (SMD).

**sort merge join.** A join method that first sorts the base tables by join attributes, then merges the sorted tables, and returns rows for matching attribute values.

**SPJ.** See stored procedure in Java (SPJ).

**SPJ method.** The static Java method associated with a stored procedure in Java (SPJ).

**SQL.** Structured query language. A non-procedural relational database language used to define, manipulate, and control databases. SQL statements can be embedded in programs or entered as commands through MXCI. See also statement atomicity.

**SQL compile.** To use the SQL compiler to generate an SQL module.

**SQL compile time.** The date and time when a program is explicitly compiled by the SQL compiler (mxcmp). Contrast with SQL statement execution time.

**SQL compiler.** The SQL/MX component that compiles a set of SQL source statements extracted from a source program by the SQL preprocessor (for C and COBOL programs) or by the SQLJ translator and SQLJ customizer (for Java programs), generates SQL object code for each statement, forms a plan for executing each statement against the database, and stores the code and plan in an SQL module. The SQL compiler also processes SQL statements entered dynamically through MXCI.

**SQL conversational interface (SQLCI).** See SQLCI.

**SQL declare section.** A section within a C or COBOL program used to declare host variables. An SQL declare section begins with BEGIN DECLARE SECTION and ends with END DECLARE SECTION.

**SQL descriptor area.** An area that contains multiple item descriptor areas, together with a count of the number of areas. An item descriptor area is used to store information about each input parameter and output variable in a dynamic SQL statement in a C or COBOL program.

**SQL diagnostics area.** An area that contains completion and exception information for the execution of an embedded SQL statement in a C or COBOL program. At the beginning of execution, the diagnostics area is emptied. When the statement executes, SQL/MX places information about completion or exception conditions into this area.

**SQL executor.** See executor.

**SQL identifier.** A name used to identify a table, view, column, and constraint, user-defined routine, or parameter to a user-defined routine. There are two types of identifiers: regular and delimited. Delimited identifiers are enclosed within double quotation characters and can include any character in the SQL-92 special character set. An identifier of either type can contain up to 128 characters.

**SQL message code.** An integer that uniquely identifies an SQL/MX error or warning. The value of this code is returned in the SQLCODE variable in an embedded SQL application. See the *SQL/MX Messages Manual* for a list of SQL message codes. See also SQLCODE.

**SQL module.** See module.

**SQL module definition file.** See module definition file (MDF).

**SQL object.** A database entity that is created, manipulated, or dropped by SQL statements and that is described in system metadata tables. SQL/MP objects are tables, views, indexes, collations, and constraints. SQL/MX objects are tables, views, indexes, constraints, locks, triggers, the trigger's temporary table, stored procedures, and SQL/MP aliases.

**SQL optimizer.** See optimizer.

**SQL parameter.** A name specified in a prepared embedded SQL statement in a C or COBOL program or in a prepared MXCI or SQLCI command for which you substitute a value when executing the statement or command. A parameter has a data type. See also dynamic input parameter and MXCI parameter.

**SQL preprocessor.** The SQL/MX component that processes a C, C++, or COBOL language source program with embedded SQL statements for subsequent compilation and execution. The preprocessor generates two output files: a host language program

with the SQL statements converted to comments and an SQL module definition file. These two files are submitted to the host language compiler and to the SQL compiler respectively.

**SQL program.** A host-language program containing embedded SQL statements to be compiled by the SQL compiler.

**SQL source file.** A file written in C, C++, COBOL, or Java, containing both SQL and programming language constructs.

**SQL statement execution time.** The time when an individual SQL statement in an SQL program executes. Contrast with SQL compile time.

**SQLCI.** SQL/MP conversational interface. SQLCI enables a user to execute SQL/MP statements and commands, and operate database utilities without embedding SQL in an application. See also MXCI.

**SQLCODE.** A value returned to MXCI or an embedded SQL program to indicate the results of executing an SQL statement. A program can use conditional statements to check the SQLCODE value to determine the results of execution. The SQLCODE variable is an integer message code specific to SQL/MX and does not comply with the ANSI standard. See also SQL message code and SQLSTATE.

**SQLJ.** Also referred to as SQLJ Part 0, the “Database Language SQL—Part 10: Object Language Bindings (SQL/OLB)” part of the ANSI SQL-2002 standard that allows static SQL statements to be embedded directly in a Java program.

**SQLJ customizer.** Uses the profile (`.ser` file) and metadata information of each embedded SQL statement in the profile to generate the module definition file (`.m` file) and extends the profile with run-time information.

**SQLJ translator.** Translates, or precompiles, the SQLJ source file (`.sqlj` file) into the Java source file (`.java` file) and the profile (`.ser` file).

**SQLSTATE.** A value returned to MXCI or an embedded SQL program to indicate the results of executing an SQL statement. A program can use conditional statements to check the SQLSTATE value to determine the results of execution. The SQLSTATE variable complies with the ANSI standard and is a five-character string with two parts: the first part is a two-character class code, and the second part is a three-character subclass code. See also SQLCODE.

**SQL-92.** The American National Standards Institute (ANSI) version of SQL (structured query language) that describes a purely relational database model. SQL-92 has been replaced with SQL:1999.

**SQL:1999.** The American National Standards Institute (ANSI) version of SQL (structured query language) that describes an object-relational database model. SQL:1999 is an updated version of the SQL-92 standard. SQL/MX uses SQL:1999 to define, manipulate, and control its databases.

**SQL:2003.** The American National Standards Institute (ANSI) version of SQL (structured query language) that includes additional object-oriented features. SQL:2003 is the latest version of the standard. SQL:2003 enables developers to apply SQL databases to problems that are too complex to address with the older, purely relational paradigm.

**SQL/MP.** The NonStop relational database management system (RDBMS) that was first introduced in 1987. SQL/MP includes system metadata; the SQL language for definition, retrieval, manipulation, and control of data; a programmatic interface; an SQL conversational interface (SQLCI); and a set of utilities.

**SQL/MP alias.** A mapping from an ANSI-defined name to the physical names of an SQL/MP table or view. The SQL/MP alias name is a three-part alias name and must contain catalog and schema names that represent actual SQL/MX ANSI catalogs and schemas. The object name for an SQL/MP alias identifies the object. The alias information for each schema is stored in the SQL/MX metadata for that schema. See also catalog, schema, and object.

**SQL/MP conversational interface (SQLCI).** See SQLCI.

**SQL/MP database.** A database created by SQL/MP that can be accessed by the SQL/MP or SQL/MX engine. Each database is a collection of tables containing data, all objects that depend on the tables, all objects that the tables depend on, all metadata in which the tables and objects are described, and all associated file labels. More than one database can be on a node, and each database can span more than one node.

**SQL/MX.** The NonStop relational database management system (RDBMS) that promotes efficient online access to large distributed databases. SQL/MX includes user metadata; the SQL language for retrieval, manipulation, and control of data; a programmatic interface; an SQL conversational interface (MXCI); and a set of utilities. Release 1 of SQL/MX can access an SQL/MP database, Release 2 of SQL/MX can access SQL/MP and SQL/MX databases.

**SQL/MX administrative command library (MACL).** See MACL.

**SQL/MX catalog manager.** The SQL/MX component that manages the metadata and physical structures of database objects. The SQL/MX catalog manager performs DDL statements that create, alter, and drop SQL/MX objects. It is also responsible for granting and revoking security privileges on SQL/MX objects.

**SQL/MX connectivity services (MXCS).** See MXCS.

**SQL/MX conversational interface (MXCI).** See MXCI.

**SQL/MX executor.** See executor.

**SQL/MX language manager.** An application programming interface (API) that the SQL/MX UDR server calls to load, invoke, and unload a stored procedure in Java (SPJ).

**SQL/MX module.** See module.

**SQL/MX optimizer.** See optimizer.

**SQL/MX software version (MXV).** The version of the SQL/MX system software that is installed on a node.

**SQL/MX UDR server.** A layer of software between the SQL/MX executor and a Java Virtual Machine (JVM) that provides support for a stored procedure in Java (SPJ) in SQL/MX. Also refers to a running instance of the MXUDR program in which SPJ methods are executed by an embedded Java Virtual Machine (JVM).

**statement atomicity.** The ability to undo the effects of an insert/update/delete (IUD) operation, when an error occurs during the operation, without having to abort the entire transaction.

**statement pinning.** The ability of SQL/MX to hold entries in the query plan cache that you want to reuse.

**static SQL.** A programming technique that uses SQL statements that are constructed and compiled before the program begins executing. Contrast with dynamic SQL.

**static SQL compilation.** See explicit SQL compilation.

**storage key.** See clustering key.

**stored procedure.** A procedure registered with SQL/MX and invoked by SQL/MX during execution of a CALL statement. A stored procedure may pass values back to its caller by assigning those values to parameters having the OUT parameter mode or INOUT parameter mode. See also stored procedure in Java (SPJ).

**stored procedure in Java (SPJ).** A stored procedure whose body is a static Java method. The body is known as an SPJ method.

**stream access.** Allows applications to access tables as continuous data streams. The stream access mode first does a regular scan of the table. Then, if all available rows have been retrieved, the stream access mode causes fetch operations to wait instead of returning the end-of-data condition.

**string literal.** Data composed of characters. A string literal has character data type.

**structured query language (SQL).** See SQL.

**subquery.** A query expression enclosed in parentheses; its syntactic form is specified in the syntax of a SELECT statement. A subquery can specify the comparison values for a predicate in a search condition or can be used as a table reference. A scalar subquery returns a single value, a row subquery returns a single row, and a table subquery returns multiple rows.

**subscribe.** To request access to entries in a queue or channel by using a SELECT statement specifying stream access.

**super ID.** A special user ID defined for each node that has NonStop operating system user name SUPER.Super and user ID 255, 255. The super ID can act as the owner of any object or file on the node.

**swap file.** A disk file used by the system for virtual memory.

**SYSKEY.** A storage key defined by the system rather than by the user. See also clustering key.

**SYSTEM_DEFAULTS.** A user metadata table that contains default settings for options and other attributes that affect the execution of commands and SQL queries that can be run through MXCI or within an embedded SQL application.

**system-defined transaction.** A TMF transaction initiated by SQL/MX either in a program unit or in a requester, on whose behalf the program unit performs database operations. Contrast with user-defined transaction. Most DDL and DML statements are transaction-initiating; the system automatically initiates a transaction when the statement begins executing. The exceptions include DML statements executing on nonaudited tables or under read uncommitted access on audited tables.

**system metadata (SMD).** Definition schemas and system schemas that contain SMD tables. Contrast with user metadata (UMD).

# T

**table.** A logical representation of data in a database in which a set of records is represented as a sequence of rows, and the set of fields common to all the records is represented as a series of columns. The intersection of a row and column represents the data value of a particular field in a particular record. As a database object, a table defines data in columns and defines the physical characteristics of the table. Contrast with file.

**table lock.** A lock held by a process on all the rows in all partitions of a partitioned table. Contrast with partition lock and row lock.

**table reference.** A reference in the FROM clause of a SELECT statement that can be a table, view, derived table, or joined table, and that determines the contents of an intermediate result table from which SQL/MX returns the columns you specify in the select list.

**targeting.** The ability to alter which set of SQL tables is operated on by an application, without changing the application's source code. In SQL/MX this is handled by late name resolution. The name can be resolved at runtime or when the application is SQL-compiled. The application approach is useful for production environments.

**timeout.** A method used by SQL/MX to alleviate deadlocks. If a process cannot acquire a lock within a given period, the SQL/MX system rejects the lock request and returns a timeout error.

**timestamp check.** A comparison performed by SQL/MX of the current redefinition timestamp of a table in an SQL statement with the table's compile-time timestamp to determine if the table has changed since the last explicit SQL compilation. If the redefinition timestamp of the table has changed, SQL/MX automatically recompiles the SQL statement if it fails the similarity check (or if the similarity check is not enabled).

**TMF.** See Transaction Management Facility (TMF).

**TNS/R native.** The Tandem NonStop Series/RISC native mode that enables users to write programs that make full use of the RISC instruction set and processor architecture. TNS/R native code has significant performance advantages over TNS code. Embedded SQL/MX is available only on TNS/R native platforms.

**token.** Additional information about an operator found in the Description column of the Explain output.

**transaction.** An operation or a series of operations that transforms a database from one consistent state to another. The HP NonStop Transaction Management Facility (TMF) product treats transactions as a single unit. Either all of the changes made by a transaction are made permanent (the transaction is committed), or none of the changes are made permanent (the transaction is aborted). If an error occurs while a transaction is in progress, the TMF product backs out whatever partial changes were made to the database, leaving it in a consistent state. See also aborted transaction.

**transactional queuing.** Applications specifying stream access can read and delete entries with a single operation as part of a transaction. Entries are dequeued after the associated transaction commits. To prevent concurrent transactions—for example, multiple applications dequeuing requests—from interfering with each other, applications use the skip conflict access mode to skip conflicting locks held by concurrent transactions.

**transaction access mode.** The mode that determines whether statements in a transaction can read but cannot insert, delete, or update data in tables (READ ONLY); or whether statements in a transaction can read, insert, delete, or update data in tables (READ WRITE).

**transaction isolation level.** See isolation level.

**Transaction Management Facility (TMF).** A NonStop product that provides transaction protection and database consistency. It gives full protection to transactions that access distributed SQL/MP databases, as well as recovery capabilities for transactions and entire databases.

**translator.** See SQLJ translator.

**trigger.** A set of actions that are executed automatically whenever a delete, insert, or update operation occurs on a specified base table.

**trigger temporary table.** A pseudo-temporary table created to implement triggers efficiently. These tables are not temporary tables, as defined by ANSI, but an implementation used to help execute triggers.

**tuple.** A data object containing two or more components.

**type checking.** Determines whether data types are compatible. See also compatible data types.

**type 1 plan.** A parallel execution plan in which matching partitions exist in tables that are involved in a join.

**type 2 plan.** A parallel execution plan the optimizer achieves by using parallel access to the inner table. The optimizer may choose this type of plan when the inner and outer tables do not match and repartitioning is undesirable or not possible.

# U

**UDR.** See user-defined routine (UDR).

**UID.** See unique identifier (UID).

**UMD.** See user metadata (UMD).

**unilateral abort.** An abort caused by a participant in a transaction that is not yet in the prepared state.

**unique identifier (UID).** An identifier, used in system metadata tables, that is generated and assigned to each catalog, schema, object, and partition at the time it is created. Each UID is unique only within a node.

**unique index.** An index in which the combined values of the columns that make up the index uniquely identify the rows in the underlying base table.

**upsert.** An SQL insert operation that becomes an update operation if the target row already exists.

**user-defined routine (UDR).** A routine registered with SQL/MX and invoked by SQL/MX during execution of an SQL statement. A stored procedure is a type of UDR. See also routine, stored procedure, and stored procedure in Java (SPJ).

**user-defined transaction.** A TMF transaction initiated by an application either in a program unit or in a requester on whose behalf the program unit performs database operations. Contrast with system-defined transaction.

**user metadata (UMD).** User schemas contain, in addition to user tables, a small number of UMD tables. System schemas also contain UMD tables however, definition schemas do not contain UMD tables. Contrast with system metadata (SMD).

**user metadata tables.** For SQL/MX, these metadata tables reside in user schemas. The HISTOGRAM and HISTOGRAM_INTERVALS tables contain histogram statistics for user tables within a database. One set of these tables resides in each user schema. The SYSTEM_DEFAULTS table contains default settings for query execution options. There is one SYSTEM_DEFAULTS table on each node where SQL/MX is installed. There are additional user metadata tables, MVS_TABLE_INFO_UMD, MVS_UMD, and MVS_USED_UMD, that are currently unused but reside in each user schema. For SQL/MP objects referenced by SQL/MX, these are SQL/MP tables, called HISTOGRM and HISTINTS, created by SQL/MX that contain histogram statistics for SQL/MP tables. See also definitional integrity, host identifier, and MPALIAS table.

**utility.** A software tool that can be used to perform a specialized task. SQL/MX utilities can be command -line driven, such as Import/Export, ImportDDL/ExportDDL, and Cleanup/Verify; syntaxed-based, such as MODIFY, DUP, and PURGEDATA; or API-based, such as NSM/web, and Backup/Restore.

# V

**versioning.** To enable different versions of software to interoperate across nodes and releases. Database versioning allows the software to determine if the version of a particular object is compatible. Query plan versioning allows the software to assign a version to query plans and modules, and determine if that version is compatible.

**view.** A table that has a logical definition and a file label but contains no data. A view is derived by projecting a subset of the columns, restricting a subset of the rows, or both, from one or more base tables or other views.

**visibility.** The ability to see a database object from a node if a catalog reference for that object's catalog exists on that node.

**Visual Query Planner.** An SQL/MX application that enables you to extract and display query execution plans generated by the optimizer for DML statements. Visual Query Planner connects to an SQL/MX database by using ODBC/MX and queries the result table of the EXPLAIN function, as applied to a DML statement, to extract query plan information.

**volume recovery.** The automatic process of recovering database files to their most recent consistent state if they become inconsistent because of a disk volume or system failure. To recover files, the volume recovery process redoes committed transactions to ensure that they are reflected correctly in the database, and then backs out all transactions that were incomplete at the time of the interruption.

# W

**wild-card character.** A character that matches any other character or set of characters in a fileset, predicate, or command.

**worktable.** An internal object that is visible through one or more physical NonStop operating system files. SQL/MX worktables handle special features supported in SQL/MX, such as triggers.