IBM Informix
Version 11.70

IBM Informix Optical Subsystem Guide

SC27-3564-00

IBM Informix
Version 11.70

IBM Informix Optical Subsystem Guide

SC27-3564-00

Note:
Before using this information and the product it supports, read the information in "Notices" on page B-1.

Contents

Introduction

In This Introduction

This introduction provides an overview of the information in this publication and describes the conventions it uses.

About This Publication

This publication describes the Optical Subsystem, a configuration of IBM Informix database software that supports the storage of TEXT and BYTE data types, also known as *simple large objects*, on optical platters. It describes the Optical Subsystem interface for an optical-storage subsystem and the set of SQL statements that you use exclusively with optical platters.

New editions and product names

Dynamic Server editions were withdrawn and new Informix® editions are available. Some products were also renamed. The publications in the Informix library pertain to the following products:

- IBM® Informix database server, formerly known as IBM Informix Dynamic Server (IDS)
- IBM Informix OpenAdmin Tool for Informix, formerly known as OpenAdmin Tool for Informix Dynamic Server (IDS)
- IBM Informix SQL Warehousing Tool, formerly known as Informix Warehouse Feature

For more information about the Informix product family, go to http://www.ibm.com/software/data/informix/.

Types of Users

This publication is for the following users who want to take advantage of write-once-read-many (WORM) optical media for storage of simple large objects:

- Database administrators
- Database server administrators

This publication assumes that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts

If you have limited experience with relational databases, SQL, or your operating system, refer to *IBM Informix Getting Started Guide* for a list of supplementary titles.

Software Dependencies

This publication assumes that you are using IBM Informix software.

Assumptions About Your Locale

IBM Informix products can support many languages, cultures, and code sets. All culture-specific information is brought together in a single environment, called a Global Language Support (GLS) locale.

This publication assumes that you use the U.S. 8859-1 English locale as the default locale. The default is **en_us.8859-1** (ISO 8859-1) on UNIX platforms or **en_us.1252** (Microsoft 1252) for Windows environments. This locale supports U.S. English format conventions for dates, times, and currency, and also supports the ISO 8859-1 or Microsoft 1252 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you must specify the appropriate nondefault locale.

For instructions on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the *IBM Informix GLS User's Guide*.

Demonstration Databases

The DB-Access utility, which is provided with your IBM Informix database server products, includes one or more of the following demonstration databases:

- The **stores_demo** database illustrates a relational schema with information about a fictitious wholesale sporting-goods distributor. Many examples in IBM Informix manuals are based on the **stores_demo** database.
- The **superstores_demo** database illustrates an object-relational schema. The **superstores_demo** database contains examples of extended data types, type and table inheritance, and user-defined routines.

For information about how to create and populate the demonstration databases, see the *IBM Informix DB–Access User's Guide*. For descriptions of the databases and their contents, see the *IBM Informix Guide to SQL: Reference*.

The scripts that you use to install the demonstration databases reside in the **$INFORMIXDIR/bin** directory on UNIX platforms and in the **%INFORMIXDIR%\bin** directory in Windows environments.

Example Code Conventions

Examples of SQL code occur throughout this publication. Except as noted, the code is not specific to any single IBM Informix application development tool.

If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores_demo
...

DELETE FROM customer
   WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

To use this SQL code for a specific product, you must apply the syntax rules for that product. For example, if you are using an SQL API, you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement. If you are using DB-Access, you must delimit multiple statements with semicolons.

Tip: Ellipsis points in a code example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept being discussed.

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the documentation for your product.

Additional Documentation

Documentation about this release of IBM Informix products is available in various formats.

All of the product documentation (including release notes, machine notes, and documentation notes) is available from the information center on the Web at http://publib.boulder.ibm.com/infocenter/idshelp/v117/index.jsp. Alternatively, you can access or install the product documentation from the Quick Start CD that is shipped with the product.

Compliance with Industry Standards

IBM Informix products are compliant with various standards.

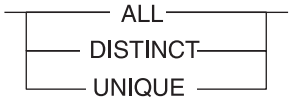
The American National Standards Institute (ANSI) and the International Organization of Standardization (ISO) have jointly established a set of industry standards for the Structured Query Language (SQL). IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of IBM Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment (CAE) standards.

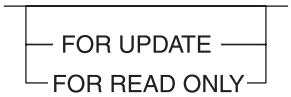
Syntax Diagrams

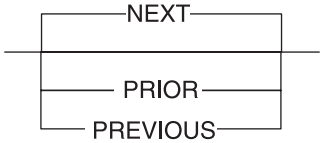
Syntax diagrams use special components to describe the syntax for statements and commands.

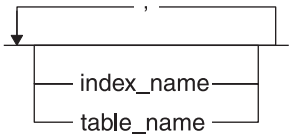
Table 1. Syntax Diagram Components

Component represented in PDF	Component represented in HTML	Meaning
	`>>----------`	Statement begins.
	`---------->`	Statement continues on next line.
	`>----------`	Statement continues from previous line.
	`----------><`	Statement ends.
	`--------SELECT----------`	Required item.
	`--+-----------------+---` `  '------LOCAL------'`	Optional item.

	`---+-----ALL-------+---`
	`   +--DISTINCT-----+`
	`   '---UNIQUE------'`
		Required item with choice. One and only one item must be present.

	`---+------------------+---`
	`    +--FOR UPDATE-----+`
	`    '--FOR READ ONLY--'`
		Optional items with choice are shown below the main line, one of which you might specify.

	`    .---NEXT---------.`
	`----+----------------+---`
	`    +---PRIOR--------+`
	`    '---PREVIOUS-----'`
		The values below the main line are optional, one of which you might specify. If you do not specify an item, the value above the line will be used as the default.

	`  .-------,-----------.`
	`  V                   |`
	`---+-----------------+---`
	`    +---index_name---+`
	`    '---table_name---'`
		Optional items. Several items are allowed; a comma must precede each repetition.

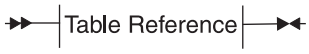
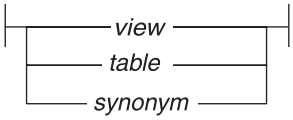
	`>>-| Table Reference |-><`	Reference to a syntax segment.

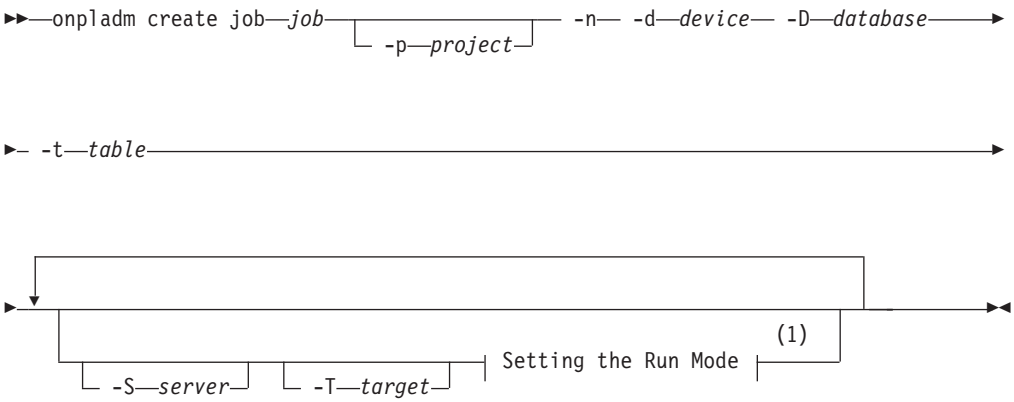
Table Reference
 
	`Table Reference`
	`|--+-----view--------+--|`
	`   +------table------+`
	`   '----synonym------'`
		Syntax segment.

How to Read a Command-Line Syntax Diagram

Command-line syntax diagrams use similar elements to those of other syntax diagrams.

Some of the elements are listed in the table in Syntax Diagrams.

Creating a No-Conversion Job

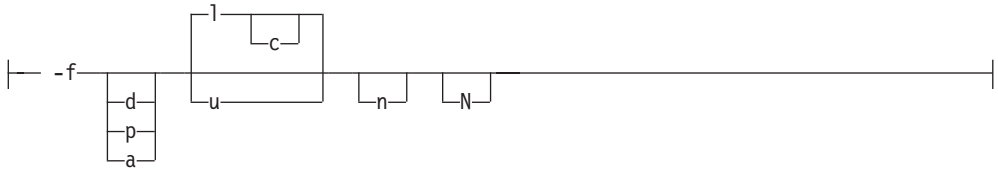


Notes:

1 See page Z-1

This diagram has a segment named “Setting the Run Mode,” which according to the diagram footnote is on page Z-1. If this was an actual cross-reference, you would find this segment in on the first page of Appendix Z. Instead, this segment is shown in the following segment diagram. Notice that the diagram uses segment start and end components.

Setting the Run Mode:



To see how to construct a command correctly, start at the top left of the main diagram. Follow the diagram to the right, including the elements that you want. The elements in this diagram are case sensitive because they illustrate utility syntax. Other types of syntax, such as SQL, are not case sensitive.

The Creating a No-Conversion Job diagram illustrates the following steps:

1. Type **onpladm create job** and then the name of the job.
2. Optionally, type **-p** and then the name of the project.
3. Type the following required elements:
   - **-n**
   - **-d** and the name of the device
   - **-D** and the name of the database
   - **-t** and the name of the table

4. Optionally, you can choose one or more of the following elements and repeat them an arbitrary number of times:
   - **-S** and the server name
   - **-T** and the target server name
   - The run mode. To set the run mode, follow the Setting the Run Mode segment diagram to type **-f**, optionally type **d**, **p**, or **a**, and then optionally type **l** or **u**.
5. Follow the diagram to the terminator.

Keywords and Punctuation

Keywords are words reserved for statements and all commands except system-level commands.

When a keyword appears in a syntax diagram, it is shown in uppercase letters. When you use a keyword in a command, you can write it in uppercase or lowercase letters, but you must spell the keyword exactly as it appears in the syntax diagram.

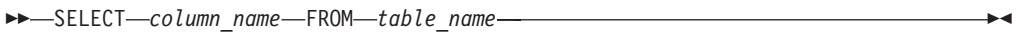
You must also use any punctuation in your statements and commands exactly as shown in the syntax diagrams.

Identifiers and Names

Variables serve as placeholders for identifiers and names in the syntax diagrams and examples.

You can replace a variable with an arbitrary name, identifier, or literal, depending on the context. Variables are also used to represent complex syntax elements that are expanded in additional syntax diagrams. When a variable appears in a syntax diagram, an example, or text, it is shown in *lowercase italic*.

The following syntax diagram uses variables to illustrate the general form of a simple SELECT statement.



When you write a SELECT statement of this form, you replace the variables *column_name* and *table_name* with the name of a specific column and table.

How to Provide Documentation Feedback

You are encouraged to send your comments about IBM Informix user documentation.

Use one of the following methods:

- Send e-mail to docinf@us.ibm.com.
- Go to the information center at http://publib.boulder.ibm.com/infocenter/idshelp/v117/index.jsp and open the topic that you want to comment on. Click the feedback link at the bottom of the page, fill out the form, and submit your feedback.
- Add comments to topics directly in the Informix information center and read comments that were added by other users. Share information about the product documentation, participate in discussions with other users, rate topics, and

more! Find out more at http://publib.boulder.ibm.com/infocenter/idshelp/v117/topic/com.ibm.start.doc/contributing.htm.

Feedback from all methods is monitored by those who maintain the user documentation. The feedback methods are reserved for reporting errors and omissions in our documentation. For immediate help with a technical problem, contact IBM Technical Support. For instructions, see the IBM Informix Technical Support website at http://www.ibm.com/planetwide/.

We appreciate your suggestions.

Chapter 1. An Overview of the Optical Subsystem

In This Chapter

The Optical Subsystem is a functionality of Informix database servers that supports the storage of TEXT and BYTE data on an optical platter.

The Optical Subsystem includes an interface for an optical-storage subsystem and supports a set of SQL statements that are exclusively for use with optical platters. These SQL statements support the efficient storage and retrieval of data to and from the optical-storage subsystem.

For more information about TEXT and BYTE data types, refer to the *IBM Informix Guide to SQL: Reference*.

Important: The Optical Subsystem does not store CLOB and BLOB data types. The optical-storage subsystem uses a file interface to support smart large objects. For more information about how to store CLOB and BLOB data types on optical disc, refer to the *IBM Informix ESQL/C Programmer's Manual*.

This chapter describes the purpose and implementation of the Optical Subsystem. The following topics are covered:

- The advantages that optical media offers
- The components of an optical-storage subsystem
- The ways in which you can organize data on optical platters
- The sequential and clustered methods of storing text and byte data on optical platters

- The operation of the Optical Subsystem application-programming interface (API), which supports the optical-storage subsystem

The Advantages of Optical Media

Optical media offer the following advantages for storing data over conventional magnetic disks:

- Mass storage capacity (on the order of gigabytes)
- Mountable/unmountable storage units
- Low cost per bit of storage
- Long media life
- High data stability

Because optical media are usually written to and read with a laser, a bit of data is written to a much smaller area on the optical *platter* (disc) than the area that is required to record a bit on a conventional magnetic disk. Optical platters offer a high degree of stability because they are far less susceptible than magnetic disks to corruption by environmental influences such as electromagnetic fields. Furthermore, the possibility of a read/write head crash is extremely remote because the optical disc read/write head does not come as close to the platter as the magnetic disk read/write head.

The Optical Subsystem supports only the write-once-read-many (WORM) type of optical media. When data is written to WORM optical media, a permanent, virtually incorruptible mark is made on the platter. Thereafter, you can read the data an unlimited number of times. When you update the data to a WORM optical platter, it is written to a new location on the platter; the optical-storage subsystem cannot reuse the original location.

Optical Media and TEXT and BYTE Data

In a database environment, the vast storage capacity of optical media make it particularly attractive for storing TEXT and BYTE data such as text documents, source and object programs, and scanned images. TEXT and BYTE data theoretically has no maximum size, although in practice the Optical Subsystem allows a maximum size of about 2 gigabytes per TEXT and BYTE data object. The number of locks available in the system might impose an additional constraint on the size.

The trade-off for the capacity, lower cost, integrity, and security that optical media provide, however, is slower access. In general, accessing optical media is slower than accessing magnetic disks. Typically, an optical drive accesses data at speeds that are 25 to 50 percent slower than a magnetic disk drive.

Components of the Optical-Storage Subsystem

The optical-storage subsystem has the following components:

- The platter
- The jukebox (autochanger) or stand-alone drive
- The shelf
- Management software for the optical-storage subsystem

The optical media are the removable optical platters that contain data.

You can keep optical platters in an automated library called a *jukebox* or *autochanger*. The jukebox is a cabinet that contains one or more optical platter drives, slots that store the optical platters when they are not mounted, and a robotic arm that transfers platters between the slots and the drives. Optical-storage subsystem commands initiate transport of the platters to and from the drives and manage the input/output (I/O) operations within the optical-storage subsystem.

Jukeboxes are not essential to the optical-storage subsystem. A supported optical-storage subsystem can consist of a stand-alone optical drive that holds only one platter or several stand-alone drives.

You store optical platters off-line on *shelves*. When a platter is known to the optical-storage subsystem but is not stored in the jukebox where it can be accessed, the platter is said to be *on the shelf*. The operator is responsible for storage and retrieval of platters to and from the shelf.

The management software for the optical-storage subsystem performs various tasks related to the optical-storage subsystem. It creates optical families and volumes and tracks information about them, chooses which optical drives to use and the optical platters to mount, issues commands to the jukebox, and reads and writes data to the optical platter.

Optical Platter Organization

In a WORM optical-storage subsystem that the Optical Subsystem supports, the optical media are organized into units of storage called the *volume* and the *family*. Each side of the platter is called a volume. An optical family is theoretically an unlimited collection of volumes, although in practice the constraints of the optical-storage subsystem impose a limit on its size. When a volume becomes full, the optical-storage subsystem automatically allocates another volume for the family.

The optical-storage administrator creates an optical family with the management software that the vendor of the optical-storage subsystem provides. The administrator also uses optical-storage subsystem commands to add volumes to a family.

In a manner that is similar to an automated tape library, the optical-storage subsystem tracks the location of each TEXT and BYTE data object on a volume, each volume on a platter, and each platter in a family. Subsystem software schedules the optical drives and mounts platters as needed.

Figure 1-1 shows the relationship between the optical family, the volume, and the optical platter.

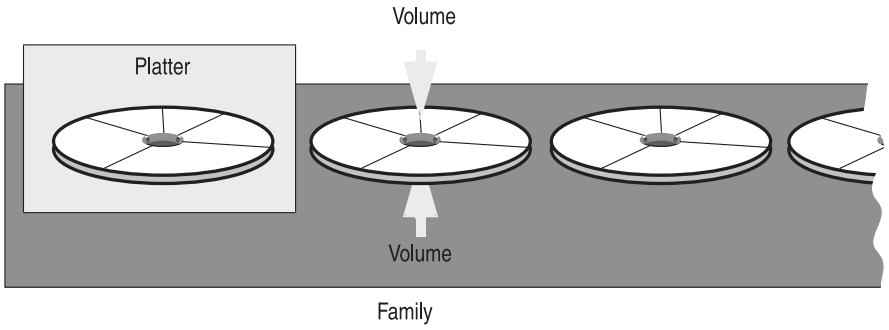


Figure 1-1. The Optical Family

Storage of TEXT and BYTE Data

You assign a TEXT or BYTE column to the optical-storage subsystem with the CREATE TABLE statement, which allows you to place the column data in an optical family. Before executing the CREATE TABLE statement, the optical-storage administrator must create an optical family name with the management software that the vendor of the optical-storage subsystem provides. You can then use the column-definition portion of the CREATE TABLE statement to assign the column to the optical family. For more information, see "Assigning TEXT and BYTE Columns to an Optical Platter" on page 3-1.

TEXT and BYTE data stored on optical media are not backed up (or archived) by the **ontape**, **onload**, and **onunload** utilities. However, the **dbexport** and **dbimport** utilities can be used to unload and load TEXT and BYTE data stored on optical media.

The Optical Subsystem can store TEXT and BYTE data on an optical volume either sequentially or in a cluster. The following sections describe each of these methods.

Storing TEXT and BYTE Data Objects Sequentially

Sequential storage means that the Optical Subsystem stores TEXT and BYTE data on the current volume of the family in the same sequence that they are inserted. When a volume becomes full, the optical-storage subsystem allocates the next volume in the family; no attempt is made to keep logically related TEXT and BYTE data objects together on the same volume or even on the same platter.

Storing TEXT and BYTE Data in a Cluster

Optical clustering attempts to store logically related TEXT or BYTE data objects together on the same volume. In applications where related TEXT or BYTE data objects are often retrieved together, optical clustering minimizes time-consuming platter exchanges on the drives. This practice improves performance.

Optical clustering does not imply that TEXT or BYTE data objects that share a particular key value are stored next to each other on the optical platter. Instead, each optical cluster represents a reservation of some number of kilobytes (the *clustersize*) on a volume. However, the reserved kilobytes and the TEXT or BYTE data objects that are eventually stored might not be contiguous. The benefit of clustering is that you can find related TEXT or BYTE data objects on the same volume far faster than you can find them on separate volumes or platters.

Figure 1-2 illustrates the difference between sequential and clustered organization. In this example, the optical platters store TEXT or BYTE data objects that are scanned photographs of merchandise. The photographs belong to the **cat_descr** column in the **stores_demo** database. If you cluster the photographs, you must cluster them by reference to some other column, such as the manufacturer's name, as photographs do not have an inherent order themselves.

When stored sequentially, the optical-storage subsystem writes the photographs to the optical platter in the same order that they are inserted in the table. If you cluster them by the manufacturer code (the cluster key), the photographs for a given manufacturer are stored together. The clustered organization reduces platter exchanges on the optical drives.

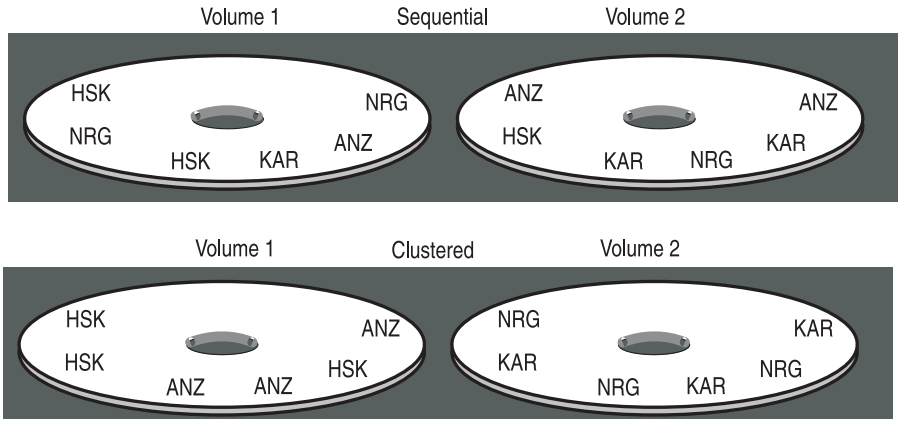


Figure 1-2. Sequential Storage Versus Clustering by manu_code

Placing TEXT and BYTE Data in Optical Clusters

You create an optical cluster with the CREATE OPTICAL CLUSTER statement. For information about the syntax, refer to "CREATE OPTICAL CLUSTER (+, DB-Access, ESQL/C)" on page 4-3. As part of the syntax to define the cluster, specify the following column lists:

column list	is one or more logically related TEXT or BYTE data columns from a table. You want to store these columns together in the same cluster.
cluster-key column list	is one or more columns that provide a logical order for the TEXT or BYTE columns. The cluster-key columns must be from the same table as the TEXT or BYTE data columns.

For example, suppose you are a sporting-goods distributor and you have photographs taken of your stock to include in an electronic catalog. With the Optical Subsystem, you create a table called **catalog**, which is similar to the **catalog** table provided with the **stores_demo** database but which stores the **cat_picture** column on an optical platter rather than a magnetic disk.

TEXT and BYTE Data Columns

In the CREATE TABLE statement for **catalog**, you define a BYTE column called **cat_picture** and use the IN portion of the column definition to place the column in the optical family **stock_photos**. For the complete syntax of the CREATE TABLE statement, see the *IBM Informix Guide to SQL: Syntax*.

The Cluster-Key Column

Next, you decide that you want to keep the photographs for a given manufacturer together to avoid unnecessary platter exchanges during retrieval. To keep photographs for a given manufacturer together, use the CREATE OPTICAL CLUSTER statement and make the **manu_code** column (also a column in the **catalog** table) the **cluster-key** column. Because **manu_code** contains nine unique values (ANZ, HRO, HSK, KAR, NKL, NRG, PRC, SHM, SMT), the photographs are stored in nine unique clusters, each containing the photographs for a single manufacturer. You assign a cluster size of 18,000 kilobytes to the cluster for the following reason: each photograph requires 60 kilobytes of storage, and you want to allow for as many as 300 photographs per manufacturer.

How Optical Clustering Occurs

Insert the first photograph into the database. As the insert operation begins, the Optical Subsystem checks whether the data is stored on an optical platter and whether an optical cluster exists for it. If so, the Optical Subsystem uses an internal optical-cluster table to find the current volume for the cluster-key value. If the cluster-key (**manu_code**) value for the first insert is HSK, the Optical Subsystem asks the optical-storage subsystem to reserve 18,000 kilobytes on the current volume for this cluster key. This first photograph is then written onto the volume.

The Optical Subsystem adds the cluster-key value, HSK, and the location of the HSK cluster to the cluster-size information that is already recorded in its internal optical-cluster table. The Optical Subsystem also records the fact that 60 kilobytes in the HSK cluster are now used. When the next unique cluster-key value, NRG for example, is inserted, the optical-storage subsystem reserves another 18,000 kilobytes on the current volume for the data associated with this cluster key. This process is repeated each time the optical-storage subsystem encounters a unique cluster-key value.

As HSK and NRG photographs fill their respective clusters, the internal optical-cluster table tracks the amount of space used in each cluster. Before a photograph is written to an optical platter, the Optical Subsystem compares the size of the object with the amount of space that remains in its assigned cluster. If the cluster is too full to receive the photograph, the optical-storage subsystem creates a new cluster on the volume.

If all space on the current volume is filled or reserved, the optical-storage subsystem finds the next available volume in the family (a stand-alone optical-storage subsystem prompts you to mount an empty platter), and creates a new cluster on it for the cluster-key value. When the new cluster is created, the internal optical-cluster table is updated with the new volume number and the new value for space that remains in the current cluster. If an additional volume is not available in the family, an error is returned.

Calculating Logical-Log Space for Optical Data

When you install the Optical Subsystem, you do not need to calculate any logical-log space for optical TEXT and BYTE data. Informix does not write optical TEXT and BYTE data to either the logical log or physical log. Optical TEXT and BYTE data is *not* written to the logical-log backup tapes when the logical logs are backed up.

For information about estimating the size of your logical log and physical log for tblspace TEXT and BYTE data, refer to your *IBM Informix Administrator's Guide*.

Support for Multiple Optical Virtual Processors

The Optical Subsystem supports multiple optical virtual processors (VPs). To execute optical I/O requests on multiple drives in parallel, configure multiple optical VPs. You can fine-tune the number of optical VPs based on your system configuration and usage.

You can set the number of optical VPs equal to or greater than the number of optical drives on a multiprocessor system. The optical VPs are not bound to a particular optical drive. An idle optical VP can service requests for I/O on any optical drive.

Determining the Number of Optical Virtual Processors

The following guidelines can help you determine the number of optical VPs to configure for your Optical Subsystem. The optimal number of optical VPs depends on the type of system usage.

Guideline One: The maximum number of optical VPs is equal to the number of clients performing optical I/O concurrently.

Guideline Two: However, if several clients are performing optical I/O concurrently, then use at least one optical VP per drive.

Scenario One: Clients are requesting writes on optical volumes in the same family. Multiple optical VPs for each drive allow additional clients to complete the initial setup while the first client is performing its I/O. The goal is to keep all the drives busy and maximize performance. If the optical objects are large, two optical VPs per drive are sufficient. If the optical objects are tiny (as in the case of signature verification), you need several optical VPs to keep the drives busy.

Scenario Two: Clients are requesting writes on the current optical volume in many different families. The write requests are randomly distributed across families. If the number of write requests is high, you would gain performance by having more optical VPs because you are more likely to get an I/O request for an already mounted platter before it is dismounted. Multiple optical VPs reduce the number of platter exchanges, a time-consuming operation.

Scenario Three: In a read-intensive environment, you should use one optical VP per optical drive.

Scenario Four: If the number of concurrent optical I/O request rates is low, a single optical VP for all optical drives might be adequate.

The Optical Subsystem API

The application programming interface (API) between the Optical Subsystem and the optical-storage subsystem consists of the following layers:

- An *internal* layer that represents the interaction between the Optical Subsystem and the API
- An *external* layer that represents the interaction between the API and the optical-storage subsystem

The optical-storage subsystem vendor provides the external layer of the interface in the form of a shared library that is linked to the Optical Subsystem. For

information about how to install the vendor library and how to link it with the Optical Subsystem, see Chapter 2, “Installation and Initial Configuration,” on page 2-1.

Figure 1-3 illustrates the relationships between the Optical Subsystem, the API, and the optical-storage subsystem.

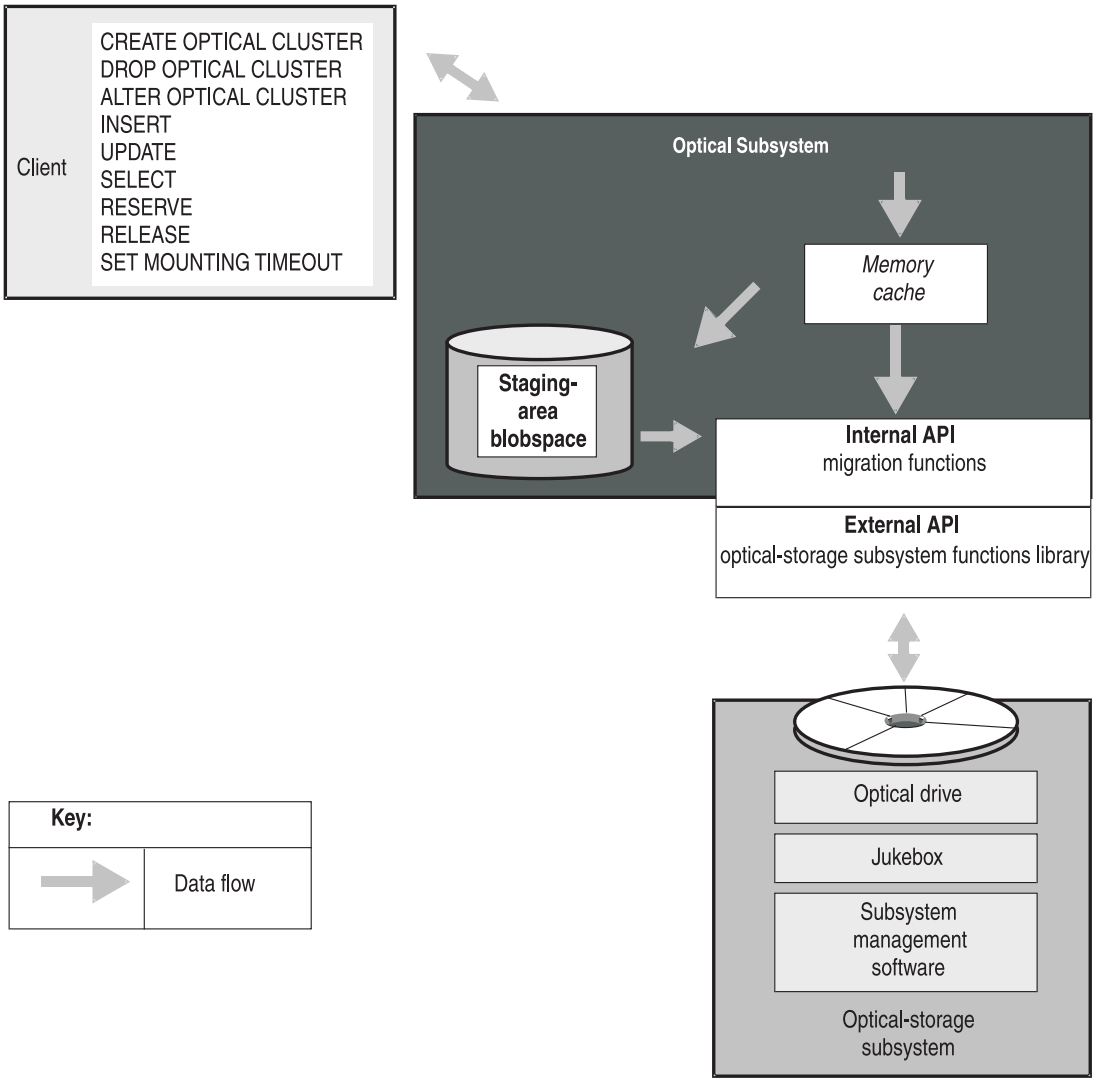


Figure 1-3. The Optical Subsystem, API, Staging-Area Blobspace, Memory Cache, and the Optical-Storage Subsystem Relationship

The Memory Cache and Staging-Area Blobspace

The staging-area blobspace improves the performance of writing TEXT and BYTE data that is larger than the cache size. The Optical Subsystem receives TEXT and BYTE data from the client application in 1-kilobyte pieces for a single row at a time. If the memory cache is full and cannot hold all pieces of TEXT and BYTE data, or if the cache is not being used, then the Optical Subsystem writes the TEXT and BYTE data to the staging-area blobspace. The Optical Subsystem uses the same memory cache for all client applications that are writing TEXT and BYTE data.

The configuration parameter OPCACHEMAX, located in the ONCONFIG file, specifies the size of the memory cache in kilobytes. If the value is `0`, the Optical

Subsystem does not use the cache. You can set the client environment variable, **INFORMIXOPCACHE**, to restrict the amount of memory cache that the client application uses. The value for **INFORMIXOPCACHE** is specified in kilobytes and can be any value less than or equal to OPCACHEMAX. If **INFORMIXOPCACHE** is not set, the client application can use as much of the memory cache as is available.

For more information about the ONCONFIG file, see the *IBM Informix Guide to SQL: Reference*. For more information about the OPCACHEMAX configuration parameter, see the *IBM Informix Administrator's Reference*.

The Optical Subsystem stores all TEXT or BYTE data for a given row in the memory cache until the last bit of TEXT or BYTE data in the row has been received or until the cache is full. If TEXT or BYTE data in the row does not fit in the memory cache, it is flushed out of the cache to the staging-area blobspace. If the next column of TEXT or BYTE data in the row fits in the available space in the memory cache, it stays in the memory cache.

For example, imagine that you have a memory cache of 150 kilobytes, that the setting for the configuration parameter OPCACHEMAX is 150 kilobytes, and that the environment variable **INFORMIXOPCACHE** is not set. Your current row has three columns of TEXT data of the following sizes: 30 kilobytes, 180 kilobytes, and 70 kilobytes. The first TEXT data column is 30 kilobytes and the Optical Subsystem writes it to the memory cache. The second TEXT data column is 180 kilobytes and the Optical Subsystem tries to use the memory cache but the data is too large. The Optical Subsystem writes the TEXT data from the second column to the staging-area blobspace. The third column of TEXT data is 70 kilobytes and fits in the memory cache. The Optical Subsystem now outmigrates all three TEXT data columns in the row to the optical-storage subsystem in 50-kilobyte pieces in the order the TEXT data columns were processed: the first TEXT data from the memory cache; the second TEXT data from the staging-area blobspace; and the third TEXT data from the memory cache.

If you want to outmigrate a different number of bytes than the default of 50 kilobytes, you can configure the outmigration amount using the storage manager for the optical-storage subsystem. Figure 1-3 on page 1-8 illustrates the use of the memory cache and the staging area when TEXT and BYTE data is outmigrated to the optical-storage subsystem.

You create the staging-area blobspace with the **onspaces** commandline utility. Before you use the optical-storage subsystem, you must perform the following tasks:

- Edit the ONCONFIG file and set the STAGEBLOB parameter to the name of the staging-area blobspace.
- Create the staging-area blobspace.
- Restart the database server to enable the Optical Subsystem to recognize the optical-storage subsystem.

Before you use the optical-storage subsystem, you might also want to perform the following tasks:

- Adjust the configuration parameter OPCACHEMAX to something other than the default (128 kilobytes).
- Specify the size for the **INFORMIXOPCACHE** environment variable in client environments.

For detailed information for each step, see "Creating the Staging Area" on page 2-2.

Memory Cache

Memory for the memory cache is allocated as it is used. If OPCACHEMAX is set at 500 (500 kilobytes) and the largest TEXT or BYTE data is 50 kilobytes, the Optical Subsystem uses only 50 kilobytes for the memory cache.

Because the setting for the configuration parameter OPCACHEMAX is system wide, 100 percent of the memory cache is available if you are the only user. If additional users compete for use of the memory cache, more data is written to the staging-area blobspace. To properly assess the use of the memory cache, look at how many applications are processing TEXT or BYTE data objects and set the OPCACHEMAX to accommodate the largest data object.

The system configuration parameter OPCACHEMAX uses a default value of 128 kilobytes; however, you can set OPCACHEMAX to any value. If OPCACHEMAX is set to `0`, the memory cache is not used and the Optical Subsystem writes all TEXT and BYTE data to the staging-area blobspace.

The Staging Area

The structure of the staging-area blobspace is the same as all other blobspaces. When it is created, the staging area consists of only one chunk, but more can be added as desired. Blobspace free-map pages manage the space. For information about how to create a blobspace, see the *IBM Informix Administrator's Reference*.

The optimal size for the staging-area blobspace depends on the following application factors:

- The frequency of data storage
- The frequency of data retrieval
- The average size of the data

To calculate the size of the staging-area blobspace, you must estimate the number of TEXT or BYTE data objects that you expect to reside in the staging-area blobspace simultaneously and multiply that number by the average size. In a single transaction, all TEXT or BYTE data that moves to the staging area is held in the staging area until the transaction completes. The number of TEXT or BYTE data objects that reside in the staging area simultaneously depends on the number of rows that are inserted at one time. The minimum size of the staging-area blobspace must be at least as large as the largest TEXT or BYTE data object that will reside in it. If the Optical Subsystem uses the staging-area blobspace, it writes an entire TEXT or BYTE data object to the staging area before it outmigrates the object to the optical-storage subsystem.

If a hardware failure occurs in the staging-area blobspace, the Optical Subsystem rolls the transaction back. The Optical Subsystem does not commit the transaction until it has written the data to the optical platter.

The staging-area blobspace cannot be mirrored.

The STAGEBLOB Parameter

You specify the stageblob blobspace with the STAGEBLOB configuration parameter. The STAGEBLOB parameter has two values separated by a comma. The first value gives the name of the blobspace and the second value gives the number of optical VPs configured. In the following example, the *blobspace_name* is the name

of the stageblob blobspace and 2 is the number of optical VPs configured. For information about configuring optical VPs, see "Support for Multiple Optical Virtual Processors" on page 1-7.

```
STAGEBLOB blobspace_name,2
```

For information about how to set the STAGEBLOB parameter, refer to the *IBM Informix Administrator's Reference*.

The presence of the STAGEBLOB parameter in the ONCONFIG file signals the database server that an optical-storage subsystem is present. If the database server does not find the STAGEBLOB parameter in the ONCONFIG file or if STAGEBLOB has a blank tag, it behaves as if no optical-storage subsystem existed. If the STAGEBLOB parameter is present but you have not created the staging-area blobspace, the database server displays the following message:

```
Invalid or missing name for Subsystem Staging Blobspace
```

You must create the staging-area blobspace and reinitialize the database server before it can migrate TEXT or BYTE data to the optical-storage subsystem. For the procedure to create the staging-area blobspace, refer to Chapter 2.

The onstat -O Option

You can use the **-O** option of the **onstat** utility to monitor available and allocated resources for the memory cache and the staging-area blobspace. Figure 1-4 shows a sample of **onstat -O** output.

```
Optical StageBlob Cache
System Cache Totals:                                  System Blob Totals:
Size    Alloc.  Avail.              Number   Kbytes  Number  Kbytes
500     500     0                   1        20      3       1500

User Cache Totals:                                    User Blob Totals:
SID     User    Size                Number   Kbytes  Number  Kbytes
94      doug    250                 1        20      1       300
95      beth    500                 0        0       2       1200
```

Figure 1-4. Example of the onstat -O Option

The example shows that Doug wrote one 20-kilobyte TEXT or BYTE data to the memory cache. He also tried to write a 300-kilobyte TEXT or BYTE data object, but it did not fit because he is limited to 250 kilobytes of the memory cache. Beth can use the entire 500-kilobyte memory cache, but the two TEXT or BYTE data objects that she tried to write did not fit in the memory cache (1,200 kilobytes) and were written to the staging-area blobspace. No optical writes are in progress in Figure 1-4 (`Avail = 0`). No optical work was completed except for these two programs because the totals equal the sum of the two user entries (1,500 kilobytes). You can tell that the two programs are still connected because the entries are visible.

The API Internal Layer

The internal layer of the API manages the following operations:

- An *outmigration* operation that stores TEXT and BYTE data in the optical-storage subsystem
- An *inmigration* operation that retrieves TEXT and BYTE data from the optical-storage subsystem

Outmigration is the operation through which the Optical Subsystem directs the optical-storage subsystem to store TEXT and BYTE data on an optical platter. Outmigration activity occurs in the following sequence:

1. The Optical Subsystem stores the TEXT or BYTE data in the memory cache in 1-kilobyte pieces.
2. The Optical Subsystem stores TEXT or BYTE data that exceeds the memory cache in the staging-area blobspace.
3. The Optical Subsystem makes the size and cluster information about the TEXT or BYTE data available to the optical-storage subsystem.
4. The optical-storage subsystem finds and reserves space to store the TEXT or BYTE data.
5. The Optical Subsystem transfers the TEXT or BYTE data to the optical-storage subsystem.
6. The Optical Subsystem signals the end of data migration. Once the outmigration process ends, the staging-area blobpages that held the migrating data are marked *free*. In addition, the memory cache is flushed and available for use.
7. The family number, volume number, and byte-offset that the optical-storage subsystem uses to describe the storage location are passed back to the Optical Subsystem, where the information is stored in the TEXT or BYTE data descriptor in the data row. (The TEXT or BYTE data descriptor resides in a database table column. The descriptor is the pointer to the actual TEXT or BYTE data on the optical-storage subsystem.)

Because data cannot be rewritten to the same location on a WORM optical platter, when the Optical Subsystem updates TEXT or BYTE data, it outmigrates it to a new, clean location on the platter and updates the descriptor to reflect the new location. The optical-storage subsystem cannot reclaim space on the WORM platter after any TEXT or BYTE data on the platter is modified or deleted.

Inmigration is the operation through which the Optical Subsystem requests and gains access to TEXT or BYTE data that is stored in the optical-storage subsystem. Inmigration activity occurs in the following sequence:

1. The Optical Subsystem requests access to TEXT or BYTE data by supplying the optical-storage subsystem with the family name, volume number, and byte offset.
2. The optical-storage subsystem locates the TEXT or BYTE data and retrieves it in pieces. It deposits each portion of the TEXT or BYTE data in memory buffers that the Optical Subsystem can access.
3. The optical-storage subsystem signals the end of optical data migration.

Memory for the Optical Transfer Buffer

The memory for the optical transfer buffer is allocated from a special pool called *opool*. The optical transfer buffer transfers BYTE and TEXT data to and from the Optical Subsystem buffer.

To track memory usage, issue the **onstat -g mem opool** command. Figure 1-5 shows the output of the **onstat -g mem opool** command.

```
Pool Summary:
          name   class  addr     totalsize  freesize  #allocfrag  #freefrag
          opool  V      a2ca018  8192       8072      1           1

          Blkpool Summary:
          name   class  addr     size       #blks
```

Figure 1-5. Example of the onstat -g mem opool Option

The API External Layer

The external layer of the API consists of a set of low-level functions that implement the interaction between the API and the optical-storage subsystem. These functions are compiled in a library that the optical-storage subsystem vendor provides.

The Optical Subsystem Message File

Under certain exceptional conditions, the Optical Subsystem API writes messages to the message file. In general, the messages indicate that the Optical Subsystem made a request of the optical-storage subsystem and the optical-storage subsystem was unable to accomplish it. For the content of these messages and an appropriate action or response, if one is required, see the documentation from your optical-storage subsystem vendor. A message consists of two error numbers. The first error number is Informix specific, and the second is generated by the vendor's subsystem log file. The MSGPATH parameter in the ONCONFIG file specifies the UNIX path name of the system message log file. For more information about the message log file, refer to the *IBM Informix Administrator's Reference*.

The database server writes the following messages concerning the Optical Subsystem to the message log file:

```
oninit: invalid or missing name for Subsystem Staging Blobspace
```

Cause: The STAGEBLOB configuration parameter specifies a blobspace, but the named blobspace does not exist. This message is normal the first time **oninit** is started with the Optical Subsystem. Check to see if you have accidentally created the STAGEBLOB as a dbspace instead of a blobspace.

Action: Use **onspaces** with the **-b** option to create the blobspace.

```
Optical Subsystem is running
```

Cause: You have set the value of the STAGEBLOB parameter in the configuration file, and the Optical Subsystem is communicating properly with the optical-storage subsystem.

Action: None required.

```
Optical Subsystem is not running
```

Cause: You have set the value of the STAGEBLOB parameter in the configuration file, but the Optical Subsystem cannot detect the existence of the optical-storage subsystem.

Or on UNIX, the **/usr/lib/iosm11a.so** link is not linked to the optical library.

Action: Check that the optical-storage subsystem is online.

```
Optical Subsystem STARTUP Error
```

Cause: The Optical Subsystem detects that the optical-storage subsystem is running, but the Optical Subsystem cannot communicate with it properly.

Action: Check your optical-storage subsystem for errors.

```
Unable to initiate communication with the Optical Subsystem
```

Cause: The optical driver that the optical-drive vendor supplies indicates that the drive is not accessible.

Action: Check the driver installation and the cabling between the computer and the drive.

Chapter 2. Installation and Initial Configuration

In This Chapter

This chapter contains instructions for installing and configuring the Optical Subsystem. It contains the following sections:

- Prerequisites lists the procedures that your optical-storage subsystem vendor must complete before you can install the Optical Subsystem.
- Supporting the Shared Library provides detailed instructions on how to connect to the support library.
- Creating the Staging Area tells you how to set the STAGEBLOB parameters in the ONCONFIG configuration file and how to create the staging-area blobspace.
- Creating Optical Families directs you to your vendor documentation for the optical-storage subsystem for instructions on how to create optical families and assign volumes to them.
- Testing the Connection directs you to perform two tests that indicate whether the connection between the Optical Subsystem and the optical-storage subsystem is operating correctly.

Prerequisites

You must obtain an optical-storage subsystem from an authorized optical-storage subsystem vendor before you can use your IBM Informix Optical Subsystem. Contact your sales representative to obtain a list of authorized optical-storage subsystem vendors.

Your vendor provides you with the support library for the optical-storage subsystem that is linked with the Optical Subsystem. You can link the Optical Subsystem to the support library during the installation process, or anytime after. Your optical-storage subsystem vendor also provides you with optical-storage subsystem utilities that perform the following functions:

- Monitor the operation of the optical-storage subsystem
- Introduce and initialize new optical platters of the optical-storage subsystem
- Perform administrative and management tasks related to the optical-storage subsystem

Your optical-storage subsystem vendor should also perform the following tasks before you install the Optical Subsystem:

- Install the optical-storage subsystem hardware
- Configure the host-operating system to support the optical-storage subsystem
- Run diagnostics to ensure proper operation of the optical-storage subsystem on the host computer

Supporting the Shared Library

The Optical Subsystem is installed automatically when you install your database server. For instructions on how to install the database server, see your *IBM Informix Installation Guide*.

If you are installing your database server for the first time, follow the instructions on how to configure and initialize the database server in your *IBM Informix Administrator's Guide*.

Edit your ONCONFIG file to set the OPTICAL_LIB_PATH to the shared optical library file used. For example, if you use Plexus optical server manager installed on the **d** drive of a Windows computer, you would set the following value in the ONCONFIG file:

```
OPTICAL_LIB_PATH d:\OSM\XDPSM\lib\libop.dll
```

Continue with Creating the Staging Area before you access the optical-storage subsystem.

Creating the Staging Area

When TEXT or BYTE data moves from the Optical Subsystem environment to the optical-storage subsystem, the data is first directed to a memory cache or a blobspace, which is a *staging area*. Before using the optical-storage subsystem, the database server administrator must perform the following tasks:

- Name the staging-area blobspace
- Initialize the database server
- Create the staging-area blobspace

For more information, refer to your *IBM Informix Administrator's Reference* and *IBM Informix Administrator's Guide*.

Naming the Staging Area on the STAGEBLOB Parameter

You must provide the name of the staging-area blobspace on the STAGEBLOB parameter.

UNIX Only

On UNIX, the blobspace name is the first field of the value of the STAGEBLOB parameter in the ONCONFIG file:

```
STAGEBLOB blobspace_name, num_optical_vps
```

End of UNIX Only

Windows Only

On Windows, the name of the staging-area blobspace is the only value of the STAGEBLOB parameter in the ONCONFIG file:

```
STAGEBLOB blobspace_name
```

End of Windows Only

Configuring Optical Virtual Processors

On UNIX, you can configure more than one optical VP. To set the number of optical VPs, use the second field of the value of the STAGEBLOB parameter. For example, if you want to use three optical VPs, specify the following values:

```
STAGEBLOB myblob,3
```

Do not leave spaces between the comma following the name of the blobspace and the number of optical VPs.

Initializing the Optical Subsystem

Initialize the database server shared memory. When you initialize the database server, the following error message appears on the screen:

```
Invalid or missing name for Subsystem Staging Blobspace
```

The Optical Subsystem is still initialized even though the staging-area blobspace has not yet been created. The staging-area blobspace name supplied as the STAGEBLOB parameter informs the database server that an optical-storage subsystem is present. You must create the staging-area blobspace, however, before you can outmigrate TEXT and BYTE data to the optical-storage subsystem.

Creating the Staging-Area Blobspace

The staging-area blobspace is the same as any other blobspace. The name of the staging-area blobspace must be the same as the name that you provided on the STAGEBLOB parameter. After you create the staging-area blobspace, you must restart the database server.

UNIX Only

On UNIX, you create the staging-area blobspace by using the **onspaces** utility.

To initialize the Optical Subsystem and create the staging-area blobspace on UNIX:

1. Name the staging-area blobspace by editing the ONCONFIG file.
2. Log in as user **informix**.
3. Initialize the database server:
   - Go to the Control Panel and double-click the **Services** icon.
   - Select your database server name in the Services window and click **Start**.
   - If you are initializing the database server for the first time, type `-iy` in the Startup Parameters window. (Use the **-iy** parameters ONLY on new systems. The **-i** parameter deletes all existing databases).
4. Wait for the database server to have a status of online before you use **onspaces**. To check the status, use **onstat**.
5. Add staging-area blobspace.
   - Change to the directory where the blobspace should be stored:

     ```
     cd pathname
     ```
   - Create the file for the blobspace:

     ```
     touch stageblob
     ```

- Make sure the blobspace has permissions 660 for user **informix**:

```
chmod 660 stageblob
onspaces -c -b stageblob -g 4 -p pathname/stageblob -o
          0 -s 10000
```

6. Start new log:

```
onmode -l
```

7. Back up logs:

```
ontape -a
```

8. Do a level-0 archive:

```
ontape -s -L 0
```

9. Stop the database server:

```
onmode -yuk
```

10. Start the database server:

```
oninit
```

End of UNIX Only

Windows Only

On Windows, you can use the **onspaces** command-line utility to create the staging-area blobspace.

To use onspaces to initialize the Optical Subsystem and create the staging-area blobspace on Windows:

1. Name the staging-area blobspace by editing the ONCONFIG file.
2. Log in as user **informix**.
3. Initialize the database server:
   - Go to the Control Panel and double-click the **Services** icon.
   - Select your database server name in the Services window and click **Start**.
   - If you are initializing the database server for the first time, type `-iy` in the Startup Parameters window. (Use the **-iy** parameters ONLY on new systems. The **-i** parameter deletes all existing databases).
4. Wait for the database server to have a status of Started before you use **onspaces**. To check the status, use **onstat**.
5. Create a file for the blobspace:

```
copy NUL c:\ifmxdata\dbservername\stageblob
```

6. Create the blobspace:

```
onspaces -c -b stageblob -g 4 -p
c:\ifmxdata\dbservername\stageblob -o 0 -s 10000
```

7. Start new log:

```
onmode -l
```

8. Back up logs:

```
ontape -a
```

9. Do a level-0 archive:

```
ontape -s -L 0
```

10. Bring down **oninit**:

```
onmode -yuk
```

11. Start the *dbservername* service:
    - Go to the Control Panel and double-click the **Services** icon.

- Select your database server name in the Services window and click **Start**.

End of Windows Only

Verifying the Presence of the Optical Subsystem

After configuration, check the message log file to make sure that your database server recognizes the Optical Subsystem. The following message should appear:

```
Optical Subsystem is running.
```

If this message does not appear, check to make sure that the setup of the Optical Subsystem is correct and that the Optical Subsystem configuration is correct.

Creating Optical Families

TEXT and BYTE data that is outmigrated to the optical-storage subsystem is stored on a volume in an optical family that you assign using the CREATE TABLE statement. Before you can assign TEXT and BYTE data to an optical family, however, you must use the utilities that the vendor of your optical-storage subsystem provides to create an optical family and assign volumes to it. For the procedure to create an optical family, see the documentation that your optical-storage subsystem vendor provides.

Testing the Connection

The following tests ensure that you followed the relink and install procedures correctly and that the connection between the Optical Subsystem and the optical-storage system is operational.

Test One

Run the optical utilities of the optical-storage subsystem vendor that monitor the operation of the optical-storage subsystem and provide the options for platter management.

The Optical Subsystem log file should indicate no abnormalities.

Test Two

Start the database server while the optical-storage subsystem is up. Both the database server message file and the log file for the optical-storage subsystem should indicate normal operations.

The message `Optical Subsystem is running` appears in the message log if the optical-storage subsystem is operating normally. For the name of the log file for the optical-storage subsystem and the messages that should appear in it, see the vendor documentation for your optical-storage subsystem.

Chapter 3. Using the Optical Subsystem

In This Chapter

This chapter documents the SQL extensions that the Optical Subsystem supports to access WORM optical media. It provides examples that show you how to perform the following functions:

- Assign a TEXT or BYTE data column to an optical platter (CREATE TABLE).
- Create an optical cluster (CREATE OPTICAL CLUSTER).
- Reserve an optical volume (RESERVE).
- Release an optical volume (RELEASE).
- Set the mounting time-out (SET MOUNTING TIMEOUT).
- Locate the optical volume where a TEXT or BYTE column is stored (FAMILY() and VOLUME() functions).
- Use TEXT or BYTE data descriptors to enable multiple tables to share the same TEXT and BYTE data (DESCR() function).
- Migrate TEXT and BYTE data in an optical family from one Optical Subsystem to another (**HPL**, **dbexport** and **dbimport**).

The examples use the **cat_descr** and **cat_picture** TEXT and BYTE columns from the **catalog** table of the **stores_demo** database. You can use the optical SQL statements from DB-Access or from a program developed with one of the IBM Informix SQL APIs, such as IBM Informix ESQL/C.

Assigning TEXT and BYTE Columns to an Optical Platter

To tell the Optical Subsystem to store a TEXT or BYTE column on an optical platter, specify an optical *family name*, in the column-definition portion of the CREATE TABLE statement. The administrator for the optical-storage subsystem must create the optical family name in the optical-storage subsystem before you can execute a CREATE TABLE statement that refers to it. For the procedure to create the optical family name, see your subsystem vendor documentation.

In the following example, the TEXT column **cat_descr** of the **catalog** table is stored with the table and the BYTE column **cat_picture** is stored in the optical family **catalog_stores_demo**:

```
CREATE TABLE catalog
     (
     catalog_num     SERIAL (10001),
     stock_num       SMALLINT NOT NULL,
     manu_code       CHAR(3) NOT NULL,
     cat_descr       TEXT IN TABLE,
     cat_picture     BYTE IN catalog_stores_demo,
     cat_advert      VARCHAR (255, 65)
     )
     IN dbspace10
```

When the statement is executed, the database server checks to determine whether **catalog_stores_demo** is a blobspace or an optical family name. Either of the following conditions returns an error:

- The name refers to both a blobspace *and* an optical family name.
- The name does not refer to either a blobspace *or* an optical family name.

Once you create the table, you can insert TEXT and BYTE data in an optical family in the following ways:

- With the LOAD statement (DB-Access) or the **dbload** utility
- From locator variables (IBM Informix ESQL/C)

Reading TEXT and BYTE Data Columns from an Optical Platter

Use the SELECT statement to read a TEXT or BYTE data column from an optical platter in the same way that you read a TEXT or BYTE data column from a magnetic disk. The syntax of the SELECT statement does not change to support access to an optical-storage subsystem. However, the Optical Subsystem supports extensions to SQL that affect how efficiently a TEXT or BYTE data column is read from the optical platter.

- Clustering TEXT and BYTE data
- Managing optical volumes

Clustering TEXT and BYTE Data

You can use the CREATE OPTICAL CLUSTER statement to store logically related TEXT and BYTE data on the same volume of an optical platter. Storing related TEXT and BYTE data together minimizes platter exchanges when the TEXT and BYTE data is retrieved. The CREATE OPTICAL CLUSTER statement requires you to perform the following steps:

- Assign a name to the optical cluster.
- Specify the TEXT or BYTE data columns in a table that are to be stored together in the optical cluster.
- Specify the cluster key.

  The cluster key consists of one or more columns in the table that define the logical grouping of the TEXT or BYTE data columns. The columns that make up the cluster key must not be TEXT or BYTE data columns.
- Specify the size of the cluster in kilobytes.

The following CREATE OPTICAL CLUSTER statement allocates a cluster of 18,000 kilobytes on an optical platter for the **cat_picture** BYTE column in the **catalog** table. It assigns the name **catalog_catpics** to the cluster.

```
CREATE OPTICAL CLUSTER catalog_catpics
   FOR catalog (cat_picture)
   ON (manu_code)
   CLUSTERSIZE 18000
```

The cluster key for **catalog_catpics** is **manu_code**. Therefore the **cat_picture** TEXT and BYTE data that have the same value in the **manu_code** column are stored together on the same volume or volumes, if more than one is required.

Choosing the Cluster Key

In the preceding example, the **manu_code** column was chosen for the cluster key because, of all the columns in the **catalog** table, it best meets the following criteria for choosing a cluster key:

- The **manu_code** column provides a logical order for retrieving the TEXT and BYTE data.
- The **manu_code** column contains duplicate values.

The first criterion for a cluster key, which can consist of more than one column, is that it provides a logical order for retrieving the TEXT and BYTE data. The purpose of the cluster key is to optimize retrieval of the TEXT and BYTE data by grouping related TEXT and BYTE data on the same volume. Grouping TEXT and BYTE data by a cluster key is only advantageous, however, if the TEXT and BYTE data is normally retrieved in the order that it is grouped. If the **cat_picture** TEXT and BYTE data is grouped by **stock_num** and then retrieved by **manu_code**, as the following example shows, the optical-storage subsystem might still need to perform multiple platter exchanges to retrieve the TEXT and BYTE data:

```
SELECT cat_picture FROM catalog WHERE manu_code = 'HRO'
```

The second criterion for a cluster key is that it must contain duplicate values. Of all the columns in the **catalog** table, only the first three, **catalog_num**, **stock_num**, and **manu_code**, provide logical sequences for accessing the table. Of these columns, **catalog_num** is not useful because it has a SERIAL data type, which produces a unique value for each row in the table. Clustering on this column produces a cluster for every row in the table where **cat_picture** is not null.

Assume, for example, that 26 rows in the **catalog** table have photographs in the **cat_picture** column, and that the **manu_code** column has seven unique values with occurrences as the following example shows:

```
manu_code      occurrences
ANZ                 7
HRO                 6
KAR                 1
NRG                 1
PRC                 4
SHM                 5
SMT                 2
```

For these characteristics, **manu_code** produces seven unique clusters.

Choosing the Cluster Size

The cluster size is the size of the cluster expressed in kilobytes. It is based on the number of TEXT and BYTE data objects that you expect to store in each cluster and the average size of the TEXT and BYTE data. You can calculate it as follows:

```
cluster size = estimated number of TEXT and BYTE data objects *
   average TEXT and BYTE data size
```

If the average size of the **cat_picture** images is 60 kilobytes, and you want to store up to 300 TEXT and BYTE data objects per cluster, the cluster size is 18,000 kilobytes.

Altering the Cluster Size

You can use the ALTER OPTICAL CLUSTER statement to change the size of an optical cluster for all clusters that are created after the statement is executed. In the following example, the size of the **catalog_catpics** cluster is reduced to 6,000 kilobytes, a cluster size that stores up to one hundred 60-kilobyte TEXT or BYTE data objects per cluster:

```
ALTER OPTICAL CLUSTER catalog_catpics CLUSTERSIZE 6000
```

The ALTER OPTICAL CLUSTER statement changes the cluster size only for TEXT or BYTE data objects created after the statement is executed. Clusters that were created previously remain their original size.

For more information, see "ALTER OPTICAL CLUSTER (+, DB-Access, ESQL/C)" on page 4-2.

Dropping an Optical Cluster

You can use the DROP OPTICAL CLUSTER statement to terminate optical clustering. The following example terminates clustering for **catalog_catpics**:

```
DROP OPTICAL CLUSTER catalog_catpics
```

The DROP OPTICAL CLUSTER statement does not affect the TEXT and BYTE data objects that were already stored in a cluster. It terminates clustering only for the affected columns in the future.

You can also use the DROP OPTICAL CLUSTER statement with the CREATE OPTICAL CLUSTER statement to change either the TEXT or BYTE data columns or the cluster-key columns for an optical cluster. For example, you can change the cluster key for **catalog_catpics** from **manu_code** to **stock_num** with the following statements:

```
DROP OPTICAL CLUSTER catalog_catpics;
CREATE OPTICAL CLUSTER catalog_catpics
   FOR catalog (cat_picture)
   ON (stock_num)
   CLUSTERSIZE 3000
```

These statements do not affect TEXT and BYTE data objects that were previously clustered on **manu_code** because you cannot alter data written to a WORM optical platter. These statements change the clustering only for future inserts or updates. If you want to recluster the TEXT and BYTE data objects that were clustered by **manu_code**, you must update them. The TEXT and BYTE data objects are then rewritten to the new cluster. The TEXT or BYTE data descriptors are updated with the new location of the TEXT and BYTE data objects. The space that these TEXT and BYTE data objects originally occupied is lost because you cannot reuse space on WORM optical media.

For more information, see "DROP OPTICAL CLUSTER (+, DB-Access, ESQL/C)" on page 4-6.

Managing Volumes on the Optical Drives

Your application can perform the following operations to manage the volumes on the optical drives and, consequently, affect the performance of the application:

- Reserve an optical volume to keep it mounted during a set of operations

- Release a reserve request to free an optical drive
- Set the time-out period for mounting an optical volume

Reserving an Optical Volume

You can use the RESERVE statement to keep a required optical volume mounted on an optical drive while you periodically access it during a set of operations. The RESERVE statement implies a request to mount the specified volume if it is not currently mounted.

When multiple users access the optical platters simultaneously, you might need to reserve a volume if your application accesses it consecutively. Otherwise, individual accesses are interleaved with requests from other applications so that each access by your application could require the volume to be remounted on the optical drive. Not reserving a volume can result in an unsatisfactory response time from the optical-storage subsystem.

For each RESERVE request, the optical-storage subsystem adds the value 1 to a reserve counter for that volume. If the value of the reserve counter is greater than zero, you cannot remove the volume. You can issue multiple RESERVE requests for a volume, but you must release each one before you can remove the volume from the drive. When you finish accessing the volume, use the RELEASE statement to release it. For details, see "Releasing a Reserved Optical Volume" on page 3-5.

If all the optical volumes mounted on the optical drives are reserved when you issue a RESERVE statement, the optical-storage subsystem waits for the number of seconds that the SET MOUNTING TIMEOUT statement specifies or for the default time-out period. For more information, see the RESERVE (+, DB-Access, ESQL/C) and SET MOUNTING TIMEOUT (+, DB-Access, ESQL/C) statements in Chapter 4, "SQL for the Optical Subsystem," on page 4-1.

You can use a SELECT statement to obtain the family name and volume number of the volume that you want to reserve. The FAMILY() and VOLUME() function expressions return the family name and volume number, respectively. In the following example, the SELECT statement uses the FAMILY() and VOLUME() functions to identify the volume that contains photographs of stock for manufacturer HRO. The RESERVE statement then reserves the volume, using the family name and volume number that the SELECT statement returned.

```
SELECT FAMILY(cat_picture), VOLUME(cat_picture)
   FROM catalog
   WHERE manu_code = 'HRO'
...
RESERVE 'catalog_1991' 013
```

Releasing a Reserved Optical Volume

You must release each RESERVE request for an optical volume with the RELEASE statement. The RELEASE statement cancels the latest RESERVE request for an optical volume by subtracting the value one (1) from the reserve counter for that volume. When the reserve counter is reduced to zero, you can remove the optical volume from the drive. The following statement releases volume 013 of the optical family:

```
RELEASE 'catalog_1991' 013
```

An error is returned if you issue a RELEASE statement for a volume that is not reserved.

The vendor of your optical-storage subsystem supplies a utility that you can also use to release a volume. The optical-storage subsystem administrator must release a volume with this utility when a program terminates before releasing a reserved volume.

Setting the Mounting Time-Out

The SET MOUNTING TIMEOUT statement controls the amount of time that your application waits for a volume to be mounted before it abandons the request. When you request access to an optical volume, the optical-storage subsystem cannot mount it immediately if all the optical drives are busy processing requests from other users. In addition, one or more drives might be reserved at the time of your request. If a SET MOUNTING TIMEOUT statement is not executed, the optical-storage subsystem times out after five minutes (300 seconds) if it cannot mount the requested volume.

The SET MOUNTING TIMEOUT statement provides the following options for setting the time-out period:

- Wait indefinitely until a drive is available for the requested volume
- Return the request for a volume immediately if it is not currently mounted on an optical drive
- Specify a number of seconds to wait for a volume to be mounted
- Wait for a volume to be mounted only if a drive is available

If it is mandatory that a particular optical volume be mounted before an application continues, it might be appropriate for the application to wait an indefinite amount of time for that volume to mount. If so, the following statement has that effect:

```
SET MOUNTING TIMEOUT TO WAIT
```

If multiple users and multiple applications are competing for the optical drives, you might not always want your application to wait for a volume that is not currently mounted. You might set the mounting time-out to NOT WAIT. For example, if your application retrieves data about an item of merchandise and the TEXT or BYTE data on an optical platter is desirable but not required, the application might not display the data if it is not immediately available.

The following example causes a request for a volume to return immediately if the volume is not currently mounted:

```
SET MOUNTING TIMEOUT TO NOT WAIT
```

Normally, you should set the time-out period to the number of seconds that is appropriate for the application to wait for a volume. The following statement sets the mounting time-out period to 30 seconds. If the volume is not mounted within 30 seconds, the process times out and control is returned to the application.

```
SET MOUNTING TIMEOUT TO 30
```

Set the mounting time-out period to zero seconds to cause the process to wait for a volume to be mounted only if a drive is currently available, as follows:

```
SET MOUNTING TIMEOUT TO 0
```

When you specify a time-out period of zero, the optical-storage subsystem waits for a volume to mount if a drive is currently available for it. When you specify the NOT WAIT option, the optical-storage subsystem does not wait for the requested volume to mount if it is not currently available on a drive.

For more information, see “SET MOUNTING TIMEOUT (+, DB-Access, ESQL/C)” on page 4-11.

Using TEXT and BYTE Data Descriptors

Data rows that include TEXT or BYTE data do not include the TEXT or BYTE data in the row itself. Instead, the data row contains a 56-byte TEXT or BYTE data descriptor that includes a pointer to the location where the first segment of data is stored. The descriptor can point to a dbspace blobpage, a blobspace blobpage, or an optical-storage location.

You can use the DESCR() function expression to retrieve the TEXT or BYTE data descriptor for TEXT or BYTE data that is stored on WORM optical media. The DESCR() function returns the descriptor in an encoded format that is 112 bytes long. Using the DESCR() function as a column value in INSERT and UPDATE statements enables you to create additional pointers to existing TEXT and BYTE data objects on the optical media. Creating additional pointers to TEXT and BYTE data objects saves space on the optical platters by allowing multiple tables to refer to the same physical TEXT and BYTE data.

Before the TEXT or BYTE data descriptor is inserted in a data row, it is decoded and validated to verify consistency and legitimacy. The optical-storage subsystem performs all encoding, decoding, and validations through the Optical Subsystem calls to API functions. If the validation process fails, the insert operation fails.

The following example uses DESCR() in an INSERT statement to insert TEXT or BYTE data descriptors, or pointers to existing TEXT and BYTE data objects, in the following hypothetical **pictures** table. Assume that the **picture** column is a BYTE column assigned to an optical platter.

```
INSERT INTO pictures (stock_num, manu_code, picture)
   SELECT stock_num, manu_code, DESCR(cat_picture)
      FROM catalog
      WHERE manu_code = 'ANZ'
```

Figure 3-1 illustrates the result of the preceding INSERT statement for a row in the **pictures** table.

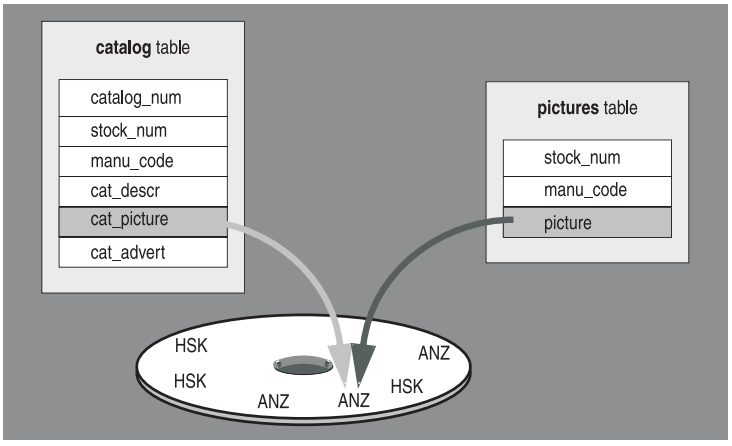


Figure 3-1. Two Descriptors That Point to the Same TEXT or BYTE Data Object on a WORM Platter

The following example uses DESCR() to update the **pictures** table with all pictures from the **cat_picture** column of the **catalog** table:

```
UPDATE pictures
   SET (picture) =
      (SELECT DESCR(cat_picture)
         FROM catalog
         WHERE pictures.stock_num = catalog.stock_num
         AND pictures.manu_code = catalog.manu_code
         AND cat_picture IS NOT NULL)
```

You can use the DESCR() function expression for TEXT or BYTE data columns that are stored on an optical platter. An error is returned if you use DESCR() on a nonoptical TEXT or BYTE data column.

Locating the Optical Volume Where TEXT and BYTE Data Is Stored

You can use the FAMILY() and VOLUME() function expressions in a SELECT statement to obtain the family name and volume number where a TEXT and BYTE data object is stored. The SELECT statement in the following example produces a list of the optical volumes that hold photographs of bicycle helmets (**stock_num** = 110):

```
SELECT FAMILY(cat_picture), VOLUME(cat_picture)
   FROM catalog
   WHERE stock_num = 110
```

Locating Columns Stored in Optical Families

To determine which columns in a database are stored on an optical platter, query the **sysblobs** table. The **sysblobs** table contains four columns that provide the following information:

- The space name (blobspace, dbspace, or family name) where the column is stored
- The media type (M = magnetic; O = optical)
- The table identification number
- The column number within the table

The following SELECT statement shows the locations of all TEXT or BYTE data columns in a given database:

```
SELECT * FROM sysblobs
```

The output that the preceding SELECT statement produces for the **stores_demo** database indicates that column number 5 in the table with tabid 109 is stored in the optical family **family1**.

```
spacename        type          tabid   colno
rootdbs            M            109      4
family1            O            109      5
```

With this information, you can query the **syscolumns** table to obtain the name of the column, as follows:

```
SELECT colname FROM syscolumns WHERE tabid = 109 AND colno = 5
```

That particular query produces the following output:

```
colname
cat_picture
```

The **oncheck** utility with the **-pD** option also tells you whether a TEXT or BYTE data column is stored on an optical platter. However, the information is displayed on a row-by-row basis**.** The following **oncheck** command displays information for rows in the **catalog** table of the **stores_demo** database**:**

```
oncheck -pD stores_demo:catalog
```

For more information about oncheck, refer to the *IBM Informix Administrator's Reference*.

Migrating a Database with TEXT and BYTE Data on an Optical Platter

When a database contains TEXT and BYTE data that is stored in an optical family, you can migrate the database from one Optical Subsystem to another using the following utilities:

- High-Performance Loader (HPL)
- **dbexport** and **dbimport**

If you are migrating a database that contains TEXT and BYTE data stored on an optical platter, check your vendor documentation for the compatibility requirements among optical-storage subsystems. For example, the destination optical-storage subsystem might need to include the same TEXT or BYTE data family name as the one that the source optical-storage subsystem uses.

The **ontape**, **onunload**, and **onload** utilities do not unload or load TEXT and BYTE data stored on optical media.

Using HPL to Unload and Load Optical Data

You can use the HPL utility to migrate optical TEXT and BYTE data. With HPL you can either load or unload a complete table or a partial table. For example, you can unload selected columns or rows that contain optical TEXT and BYTE data and insert them into a table in another database.

For instructions on how to use the HPL utility, see the *IBM Informix High-Performance Loader User's Guide*.

The dbexport and dbimport Utilities

The **dbexport** utility unloads a database into ASCII files; the **dbimport** utility creates a database from ASCII files. You can use the **-d** option of the **dbexport** utility to specify that only the TEXT or BYTE data descriptor is written for a TEXT or BYTE data column that is stored on an optical platter. Migrating the descriptor permits a TEXT and BYTE data object that is stored on an optical platter to be shared in multiple databases, eliminating the need to store duplicate copies of it on an optical platter. The TEXT or BYTE data descriptor contains information about the location and size of the data. The descriptor holds all the information that is necessary to retrieve a TEXT or BYTE data object from an optical platter. If you do not specify the **-d** option, **dbexport** exports both the descriptor and the TEXT or BYTE data object.

If you want a different cluster arrangement in the destination database and you can afford the space on an optical platter, you can choose to duplicate the TEXT and BYTE data in a different cluster.

The following **dbexport** command exports the **stores_demo** database to the **exstores_demo** directory, exporting only descriptors for tables that contain TEXT or

BYTE data columns. (The **-c** instructs the program to ignore all errors except unrecoverable errors, **-o** specifies the output directory, and **-q** suppresses the echo of SQL statements.)

```
dbexport -c -d -o exstores_demo -q stores_demo
```

TEXT or BYTE data descriptors remain valid from one Optical Subsystem to another because each volume on an optical platter contains an internal tracking label. In addition to TEXT or BYTE data, each WORM volume contains a label with the following information:

- Family name
- Family number (also referred to as the optical-storage subsystem serial number)
- Volume number
- Optical-storage-subsystem identifier

Each optical-storage-subsystem device, whether a stand-alone drive or jukebox, has a unique optical-storage-subsystem identifier. The optical-storage subsystem uses the identifier and the family name to uniquely identify the family within the optical-storage subsystem. Using the identifier and the family name becomes an important issue in a distributed database environment where more than one optical-storage subsystem might be available.

For instructions on how to use **dbexport** and **dbimport**, see the *IBM Informix Migration Guide*.

Chapter 4. SQL for the Optical Subsystem

In This Chapter

This chapter describes the syntax and usage of the individual SQL statements that the Optical Subsystem provides to support optical-storage subsystems. The syntax of each statement is illustrated in a diagram that shows the sequence of the required and optional elements. For a description of the conventions used in these diagrams, see "Syntax Diagrams" on page viii of the Introduction. The SQL statements presented in this chapter are used exclusively with optical-storage subsystems. For additional information about the context in which each statement is used, see Chapter 3, "Using the Optical Subsystem," on page 3-1.

This chapter describes the following SQL statements:

- ALTER OPTICAL CLUSTER
- CREATE OPTICAL CLUSTER
- DROP OPTICAL CLUSTER
- RELEASE
- RESERVE
- SET MOUNTING TIMEOUT

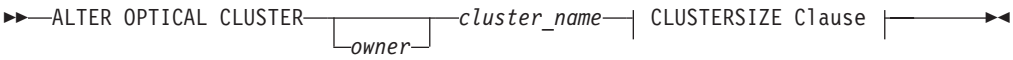
This chapter also describes the following function expressions:

- DESCR()
- FAMILY()
- VOLUME()

ALTER OPTICAL CLUSTER (+, DB-Access, ESQL/C)

Use the ALTER OPTICAL CLUSTER statement to alter the size of an optical cluster.

Syntax



CLUSTERSIZE Clause:

Element	Purpose	Restrictions	Syntax
cluster_name	The name of the cluster to alter	The cluster name must already exist.	Identifier segment; see *IBM Informix Guide to SQL: Syntax*
clustersize	Size of the cluster, in kilobytes, to be reserved for each unique cluster-key value	Clustersize must be less than the volume size.	Literal Number segment; see *IBM Informix Guide to SQL: Syntax*
owner	The user name of the owner of the cluster	The specified name must be a valid user name.	Identifier segment; see *IBM Informix Guide to SQL: Syntax*

Usage

To alter an optical cluster, you must be the owner of the cluster, have the Index privilege on the table, or have the DBA privilege.

In the following example, the CREATE OPTICAL CLUSTER statement creates **cat_clstr**, an optical cluster of 6,000 kilobytes for the **cat_picture** column in the **catalog** table. For each unique value of **stock_num** in the table, **cat_clstr** stores the associated data for this column. The following example changes the amount of space allocated for each **cat_clstr** from 6,000 kilobytes to 8,000 kilobytes. This change affects all clusters created after the ALTER OPTICAL CLUSTER statement is executed. Clusters created before an ALTER OPTICAL CLUSTER statement maintain their original size.

```
CREATE OPTICAL CLUSTER cat_clstr
     FOR catalog (cat_picture)
     ON (stock_num
     CLUSTERSIZE 6000
ALTER OPTICAL CLUSTER cat_clstr CLUSTERSIZE 8000
```

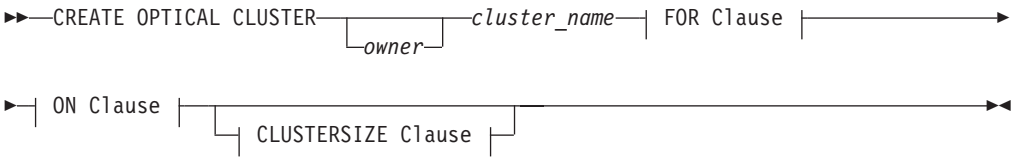
Related Information

Related statements: CREATE OPTICAL CLUSTER and DROP OPTICAL CLUSTER

CREATE OPTICAL CLUSTER (+, DB-Access, ESQL/C)

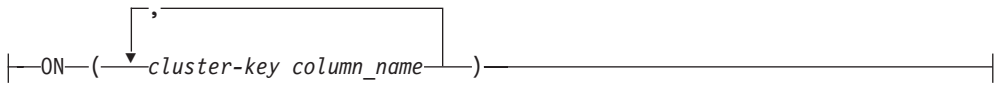
Use the CREATE OPTICAL CLUSTER statement to create a cluster on an optical platter for one or more logically related TEXT or BYTE data columns.

Syntax

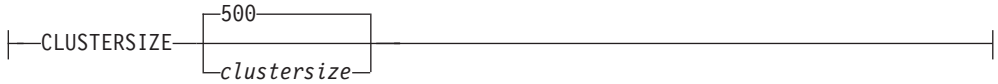


FOR Clause:

ON Clause:



CLUSTERSIZE Clause:



Element	Purpose	Restrictions	Syntax
cluster-key column_name	The name of the column that is the cluster key	You can specify up to 16 columns to create a composite cluster key. You cannot specify a TEXT or BYTE data column as part of a cluster key. The maximum length of a cluster key is 390 bytes.	Identifier segment; see *IBM Informix Guide to SQL: Syntax*
cluster_name	The name of the optical cluster	The name must be unique.	Identifier segment; see *IBM Informix Guide to SQL: Syntax*
clustersize	The amount of optical platter space to allocate for the cluster, specified in kilobytes If no clustersize is specified, the size of the cluster defaults to 500 kilobytes.	The clustersize must be less than the volume size.	Expression segment; see *IBM Informix Guide to SQL: Syntax*

Element	Purpose	Restrictions	Syntax
column_name	The name of the TEXT or BYTE data column(s) to be clustered	You can assign up to 16 distinct columns to a cluster.	Identifier segment; see *IBM Informix Guide to SQL: Syntax*
owner	The valid user name of the cluster	The specified name must be a valid user name.	Identifier segment; see *IBM Informix Guide to SQL: Syntax*

Usage

An optical cluster enables logically related TEXT or BYTE data to be stored on the same volume, which minimizes time-consuming platter exchanges during retrieval.

You can create (and, by default, become owner of) an optical cluster if you are the owner of the table, have the Resource privilege on the database and the Index privilege on the table, or have the DBA privilege on the database.

Only a person with DBA privileges can create the optical cluster and specify another user as the owner of the cluster.

TEXT and BYTE Data Columns

The following restrictions are placed on the TEXT or BYTE data columns that are being clustered on the optical platter:

- All columns within a single clustering strategy must reside on the same optical family.
- A single column cannot be clustered more than once.

Cluster-Key Columns

A cluster key can be a single column (for example, **stock_num**) or a composite of up to 16 columns (for example, **stock_num**, **manu_code**). You cannot specify a TEXT or BYTE data column as a cluster key. The maximum length of a cluster key is 390 bytes. An error message is returned if you submit a cluster key that is longer than 390 bytes.

You can gain better performance if you organize the columns in a composite cluster key so that the column with the fewest number of duplicate values is placed first.

You can create more than one optical cluster for the same list of cluster-key columns. This option differs from the restrictions placed on indexes, each of which you must create from a unique set of index-key columns.

Clustersize

The *clustersize* is the size, in kilobytes, of the space reserved on an optical volume for each cluster with a unique cluster-key value. The size of the cluster can be enlarged beyond this size, if necessary, but it cannot exceed the size of a volume. If you do not specify *clustersize*, the default size of the cluster is 500 kilobytes.

Example

Assume that the CREATE TABLE statement for the **catalog** table in the **stores_demo** database specifies that the columns **cat_descr** and **cat_picture** are to be stored in the same optical family. In the following example, the CREATE OPTICAL CLUSTER statement creates **cat_clstr**, an optical cluster of 6,000 kilobytes. A new **cat_clstr** is allocated to store the data for **cat_descr** and **cat_picture** each time a row is inserted with a unique value for **manu_code**.

```
CREATE OPTICAL CLUSTER cat_clstr
     FOR catalog (cat_picture, cat_descr)
     ON (manu_code)
     CLUSTERSIZE 6000
```

If the **stores_demo** database contains six unique values for **manu_code**, the statement lays the groundwork for the optical-storage subsystem to create six optical clusters, one for each value. If the table contains ten rows for one **manu_code**, then the **cat_clstr** for that value contains the **cat_descr** and **cat_picture** data for all ten rows.

To create the cluster, the optical-storage subsystem reserves 6,000 kilobytes on the current volume. If the current volume is too full to accept the space reservation, the cluster is created on the next available volume in the family. The location and size of each cluster on the **manu_code** cluster key is tracked in an internal optical-cluster table. The name of the optical cluster, the number of its TEXT and BYTE data columns, and the cluster-key column numbers are stored in the **sysopclstr** table.

Related Information

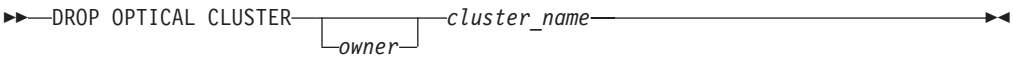
Related statements: ALTER OPTICAL CLUSTER and DROP OPTICAL CLUSTER

For more information, see the CREATE TABLE statement in *IBM Informix Guide to SQL: Syntax*.

DROP OPTICAL CLUSTER (+, DB-Access, ESQL/C)

Use the DROP OPTICAL CLUSTER statement to end optical clustering for the TEXT or BYTE data columns that are associated with the cluster name.

Syntax



Element	Purpose	Restrictions	Syntax
cluster_name	The name of the optical cluster	The cluster name must already exist.	Identifier segment; see *IBM Informix Guide to SQL: Syntax*
owner	The user name of the owner of the cluster	The specified name must be a valid user name.	Identifier segment; see *IBM Informix Guide to SQL: Syntax*

Usage

The DROP OPTICAL CLUSTER statement terminates clustering on the optical platter for the TEXT or BYTE data columns associated with *cluster name* and drops the internal optical cluster table. After the DROP OPTICAL CLUSTER statement is executed, the data items that are associated with the cluster name are stored on the current volume; they are stored in the sequence that they are output to the optical-storage subsystem.

You must be the owner of the optical cluster or have the DBA privilege on the database to drop the optical cluster.

To change the cluster-key columns of an optical cluster, drop the old optical cluster and create a new optical cluster with the new cluster-key columns. You can change your optical clustering strategy so that the **cat_picture** and **cat_descr** columns of the **catalog** table are clustered according to **stock_num** instead of **manu_code**, as follows:

```
DROP OPTICAL CLUSTER cat_clstr
CREATE OPTICAL CLUSTER cat_stock_clstr
     FOR catalog (cat_picture, cat_descr)
     ON (stock_num)
     CLUSTERSIZE 6000
```

After the new optical cluster, **cat_stock_clstr**, is created on **stock_num**, all data that is inserted into the database for the **cat_picture** and **cat_descr** columns is optically clustered according to the associated value of **stock_num**. However, the new cluster does not alter the clustering of TEXT and BYTE data already stored on the WORM optical platters. A new cluster affects only subsequent outmigrations. Therefore, in this example, the TEXT and BYTE data that **cat_clstr** previously clustered retains its original locations. In subsequent outmigrations, the data for the **cat_descr** and **cat_picture** columns is clustered by **cat_stock_clstr**, based on the value of **stock_num**. Changing the cluster key affects how the data is stored on the optical platter but not the integrity of the data.

Changing the clustering arrangement often can have an adverse affect on how efficiently data is retrieved. The TEXT and BYTE data that remains clustered by **manu_code** is effectively unclustered when **stock_num** accesses it. Accessing it by

stock_num can result in frequent platter exchanges. To alleviate this problem, update the TEXT and BYTE data that **manu_code** previously stored so that it is rewritten in the new **stock_num** clusters. Because the optical media is WORM, however, the space where the TEXT and BYTE data was stored originally cannot be reused.

Administrators and programmers should be aware that the optical clustering strategy significantly affects the use of space on the optical platters.

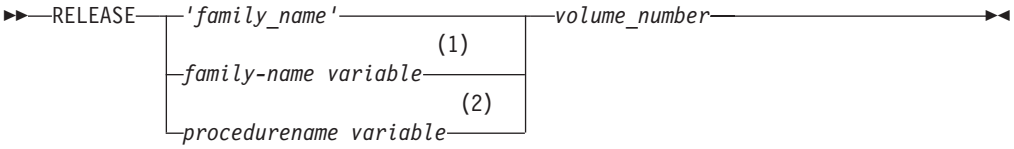
Related Information

Related statements: CREATE OPTICAL CLUSTER

RELEASE (+, DB-Access, ESQL/C)

Use the RELEASE statement to cancel a reserve request for an optical volume.

Syntax



Notes:

1 ESQL/C Only

2 SPL Only

Element	Purpose	Restrictions	Syntax
family_name	A quoted-string constant that specifies a family name in the optical-storage subsystem	Must be an existing family name.	Quoted String segment; see the *IBM Informix Guide to SQL: Syntax*.
family-name variable	A CHARACTER or VARCHAR host variable that contains a family name	Must be an existing variable name.	Variable name must conform to language-specific rules for variable names.
procedure-name variable	The name of a variable defined in an SPL routine	Must be defined in the procedure and valid in the statement block.	Identifier segment; see the *IBM Informix Guide to SQL: Syntax*.
volume_ number	The volume being released; specified as an integer	Must be reserved and mounted correctly.	Number must be an integer.

Usage

Each time that a RELEASE statement is executed, the value 1 is subtracted from the reserve counter for the volume. When the value of the reserve counter is zero, you can unmount the volume.

The volume specified in the RELEASE statement must be reserved and the platter that contains the volume must be mounted currently. An error is returned if the platter is not mounted or if the value of the reserve counter for the volume is zero.

You can issue multiple reserve requests, but you must release each one before you can remove the volume from the optical drive.

The following example releases a reservation for volume 013 in the **catalog_1991** optical family:

```
RELEASE 'catalog_1991' 013
```

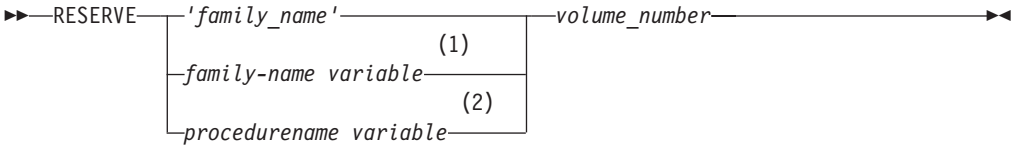
Related Information

Related statements: RESERVE

RESERVE (+, DB-Access, ESQL/C)

Use the RESERVE statement to keep a particular optical volume mounted on the optical drive for the duration of a set of operations. The RESERVE statement implies a request to mount the specified volume if it is not currently mounted.

Syntax



Notes:

1 ESQL/C Only

2 SPL Only

Element	Purpose	Restrictions	Syntax
family name	A quoted-string constant that specifies a family name in the optical-storage subsystem	Must be a unique name.	Quoted String segment; see the *IBM Informix Guide to SQL: Syntax*.
family-name variable	A CHARACTER or VARCHAR host variable that contains a family name	Must be a unique variable name.	Variable name must conform to language-specific rules for variable names.
procedure-name variable	The name of a variable defined in an SPL routine	Must be defined in the procedure and valid in the statement block.	Identifier segment; see the *IBM Informix Guide to SQL: Syntax*.
volume_ number	The number of the volume being reserved; specified as an integer	None.	Number must be an integer.

Usage

If the platter that contains the specified volume is not currently mounted on a drive, the optical-storage subsystem waits for a drive to become available and mounts the platter before it continues. The default waiting period for a mount attempt is five minutes (300 seconds). You can change this default period with the SET MOUNTING TIMEOUT statement.

You can issue multiple reserve requests for the same volume. Each time a RESERVE statement is executed for a volume, the value one (1) is added to the reserve counter for that volume. If no reserve requests have been issued for an optical volume, the value of the reserve counter is zero, indicating that you can remove the volume from the optical drive.

When you finish accessing the reserved volume, you can release it with the RELEASE statement. Each time a RELEASE statement is executed, the value 1 is subtracted from the reserve counter for that volume. Thus, you can issue multiple reserve requests, but you must release each one before you can remove the volume from the optical drive.

You can use a SELECT statement with the FAMILY() and VOLUME() functions to obtain the family name and volume number of the volume that you want to reserve. Then, you can specify that optical volume in the RESERVE statement, as follows:

```
SELECT FAMILY(cat_picture), VOLUME(cat_picture)
     WHERE manu_code = HRO AND stock_num = 301
...
RESERVE 'catalog_1991' 013
```

Related Information

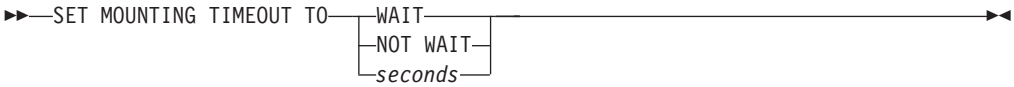
Related statements: RELEASE

SET MOUNTING TIMEOUT (+, DB-Access, ESQL/C)

When you request access to an optical volume, the optical-storage subsystem cannot mount it immediately if all the optical drives are busy processing requests from other users. In addition, one or more drives might be reserved at the time of your request.

Use the SET MOUNTING TIMEOUT statement to control the amount of time that your application waits for a volume to be mounted before it abandons the request to continue processing. The mounting time-out disregards the amount of time spent waiting for an optical VP to become available. If the SET MOUNTING TIMEOUT statement is not executed, the default value for a mounting time-out is five minutes (300 seconds).

Syntax



Element	Purpose	Restrictions	Syntax
seconds	The maximum number of seconds the process should wait for access to a volume No default value exists for time-out within the statement.	Must be specified as a positive integer or zero. Mounting time of zero seconds is the same as NOT WAIT.	Literal Number segment; see *IBM Informix Guide to SQL: Syntax*.

Usage

If you set the time-out value to WAIT, all requests for a volume wait until the volume is mounted on an optical drive. The process that made the request waits indefinitely.

If you set the time-out value to NOT WAIT, all requests for a volume return immediately if that volume is not currently mounted on an optical drive.

If you set the time-out value to a positive integer, the integer specifies the number of seconds that the process waits for a volume to be mounted before the request returns back.

If you set the time-out value to zero, the process that made the request will wait for a volume to be mounted only if an optical drive is currently available. If all the drives contain either a reserved volume or a volume that is being accessed, the request returns back. If a drive is available, the process waits for the requested volume to be mounted. In general, mounting a volume takes about 15 seconds.

The following example sets the mounting time-out period to 45 seconds. When the application subsequently requests that a volume be mounted, it waits for up to 45 seconds for that volume to be mounted. If that volume cannot be mounted within the 45-second period, the process times out.

```
SET MOUNTING TIMEOUT TO 45
```

Function Expressions (+, DB-Access, ESQL/C)

The following function expressions support the optical-storage subsystem. Use them with the SELECT statement.

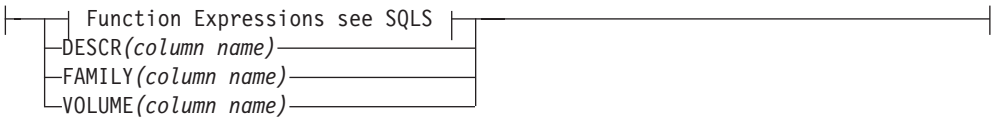
- DESCR()
- FAMILY()
- VOLUME()

You can use a function expression to perform various operations on column data or to obtain information from the database server about the contents of one or more columns. A function expression requires an argument or parameter. For all three of these function expressions, the argument is the name of a column that is stored on an optical platter. For more information about function expressions and a complete list of other function expressions, see the *IBM Informix Guide to SQL: Syntax*.

Use the DESCR(), FAMILY(), and VOLUME() function expressions to obtain information about TEXT or BYTE data columns that are stored on optical platters.

Syntax

Function Expressions:



Element	Purpose	Restrictions	Syntax
column name	The name of the TEXT or BYTE data column on which the function operates	The column must be stored on an optical platter.	Identifier segment; see *IBM Informix Guide to SQL: Syntax*.

Usage

The DESCR() function returns TEXT data that is the encoded descriptor for the TEXT or BYTE data column. The FAMILY() function returns the name of the optical family in which a column is stored. The VOLUME() function returns the number of the volume on which a column is stored. All three functions are valid *only* for data stored on an optical platter. They are valid whether the TEXT and BYTE data is stored sequentially or in clusters. An error message is returned if the functions are used on a column that is not stored on an optical platter.

DESCR()

The DESCR() function returns TEXT data, which is the encoded descriptor for the TEXT or BYTE data column. The encoded descriptor is 112 bytes. Use it as a column value in INSERT and UPDATE statements as a way to create additional pointers to existing TEXT and BYTE data on the optical platter. Before the descriptor is inserted in a data row, it is decoded and validated to verify its consistency and legitimacy. The optical-storage subsystem performs all encoding, decoding, and validations through calls by the Optical Subsystem to API functions. If the validation fails, the insert operation fails.

The DESCR() function is designed for WORM optical media only.

In the following example, the DESCR() function is used in an INSERT statement to fill the **picture** column of the **pictures** table with pointers to existing TEXT and BYTE data from the **catalog** table:

```
INSERT INTO pictures (stock_num, picture)
   SELECT stock_num, DESCR(cat_picture) FROM catalog
   WHERE manu_code = 'HRO'
```

FAMILY()

The FAMILY() function returns the name of the optical family in which a TEXT or BYTE data column is stored.

In the following example, the FAMILY() function obtains the *family name* of the optical platters that contain the data for the **cat_picture** column:

```
SELECT FAMILY(cat_picture) FROM catalog WHERE manu_code = 'HRO'
```

One row of output (*family name*) is generated for each row where **manu_code** is equal to HRO.

VOLUME()

The VOLUME() function returns the number of the volume where a TEXT or BYTE data column is stored.

Volumes are filled numerically. Once a volume is filled, or no room is available for a cluster, all subsequent writes occur on volumes with a higher number. When TEXT and BYTE data is stored sequentially, the highest volume number contains the most recently stored objects. However, this generalization is not true for clustered TEXT or BYTE data.

In the following example, the VOLUME() function generates a list of the volumes that contain data for the **cat_picture** column:

```
SELECT VOLUME(cat_picture) FROM catalog
```

Related Information

For more information about how to use DESCR(), FAMILY(), and VOLUME(), see Chapter 3.

For the following information, see the *IBM Informix Guide to SQL: Syntax*:

- A complete list of other function expressions
- The syntax of the column-definition portion of the CREATE TABLE statement, which is used to assign a TEXT or BYTE data column to an optical family
- The SELECT statement

Appendix. Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

Accessibility features for IBM Informix

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility Features

The following list includes the major accessibility features in IBM Informix. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers.
- The attachment of alternative input and output devices.

Tip: The IBM Informix Information Center and its related publications are accessibility-enabled for the IBM Home Page Reader. You can operate all features using the keyboard instead of the mouse.

Keyboard Navigation

This product uses standard Microsoft Windows navigation keys.

Related Accessibility Information

IBM is committed to making our documentation accessible to persons with disabilities. Our publications are available in HTML format so that they can be accessed with assistive technology such as screen reader software. The syntax diagrams in our publications are available in dotted decimal format. For more information about the dotted decimal format, go to "Dotted Decimal Syntax Diagrams."

You can view the publications for IBM Informix in Adobe Portable Document Format (PDF) using the Adobe Acrobat Reader.

IBM and Accessibility

See the *IBM Accessibility Center* at http://www.ibm.com/able for more information about the commitment that IBM has to accessibility.

Dotted Decimal Syntax Diagrams

The syntax diagrams in our publications are available in dotted decimal format, which is an accessible format that is available only if you are using a screen reader.

In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), the elements can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read punctuation. All syntax elements that have the same dotted decimal number (for example, all syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, the word or symbol is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is read as 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol that provides information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, that element is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to a separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

? Specifies an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element (for example, 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! Specifies a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines

2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

* Specifies a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data-area, you know that you can include more than one data area or you can include none. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

\+ Specifies a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times. For example, if you hear the line 6.1+ data-area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. As for the * symbol, you can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not

been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

> © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. (enter the year or years). All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Index

Special characters

A

B

C

D

E

T

U

V

W

Printed in USA

SC27-3564-00