StorHouse®

StorHouse Database Administration Guide

StorHouse/RM Release 3.2

Publication Number
900108 Rev. K

June 26, 2002

FileTek

All rights reserved. No part of this publication may be reproduced, translated, stored in any electronic retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of FileTek, Inc.

This publication Copyright © 1996-2002 by FileTek, Inc., Rockville, MD
Publication Number: 900108 Rev. K

NOTE: U.S. GOVERNMENT USERS

Restricted Rights Legend

Use, duplication or disclosure by the Government is subject to the restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or the Commercial Computer Software - Restricted Rights clause at 48 CFR 52.227-19, as applicable. Unpublished-rights reserved under the copyright laws of the United States. The contractor/manufacturer is:

FileTek, Inc.
9400 Key West Avenue
Rockville, Maryland 20850

Information in this document is subject to change without notice and does not represent a commitment on the part of FileTek, Inc. Further, FileTek, Inc. reserves the right to supplement the document with information not available at the time of creation of the document. FILETEK, INC. PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, AND CANNOT WARRANT THE RESULTS YOU MAY OBTAIN USING THE DOCUMENT. IN NO EVENT SHALL FILETEK, INC. BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OR DATA, INTERRUPTION OF BUSINESS, OR FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND, EVEN IF FILETEK, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES ARISING FROM ANY DEFECT OR ERROR IN THIS PUBLICATION. Some states or jurisdictions do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

FileTek and StorHouse are registered U.S. trademarks of FileTek, Inc. VRAM is a U.S. trademark of FileTek, Inc. All other brand or product names are trademarks or registered trademarks of their respective owners.

Documentation for FileTek’s StorHouse product. Protected by the following U.S. Patents: 4,864,572; 5,247,660; 5,727,197; 6,049,804. Other patents pending.

FileTek

Contents

Welcome xv

StorHouse family of products xv
StorHouse/SM xv
StorHouse/RM xvi
Control Center xvi

Purpose of this document xvi

Intended audience xvii

Contents xviii

Related documentation xix
Other StorHouse/RM publications xx
StorHouse publications xxi
Control Center publications xxi

Conventions xxii

Chapter 1: Introduction 1-1

What is StorHouse/RM? 1-1

StorHouse/RM features 1-2
Standard SQL 1-2
System managed storage 1-2
Remote database interfaces 1-2
Federation 1-3
Application program interface 1-3
Bulk data loading and unloading 1-3

Large object (LOB) support 1-3
Concurrency 1-3
Indexing 1-3
High-speed extraction 1-3
Database integrity 1-4
Controlled access/database security 1-4

StorHouse/RM software components 1-4
Data loaders 1-4
FileTek MVS Data Loader utility 1-4
FileTek FTP Data Loader 1-5
Data unloader 1-6
StorHouse engines 1-7

Chapter 2: StorHouse database concepts 2-1

StorHouse database components 2-1
Database user components 2-2
User tablespaces 2-3
User tables 2-3
User table indexes 2-4
Views 2-4
Synonyms 2-5
Database system components 2-5
System tablespace 2-5
System tables 2-6
System table indexes 2-7
System table logs 2-7
System table index logs 2-7

Segmentation 2-7
Table data files 2-9
Index files 2-9
LOB storage and subsegment files 2-10
In-line LOBs 2-11
Out-of-line LOBs 2-13
Extents 2-15

Storage management on StorHouse 2-15

Naming conventions 2-16
Database names 2-17
Database user component names 2-18
Delimited names 2-18
Unique names 2-19
Database user component identifiers 2-19
Segment and LOB subsegment identifiers 2-20
Segment tags 2-20
Segment file names 2-21
StorHouse/RM 3.2 segment file names 2-21
StorHouse/RM 2.3 segment file names 2-23
StorHouse/RM 2.1 to 2.2A segment file names 2-24
Database system component file names 2-26
Database system component identifiers 2-26

Summary of StorHouse database concepts 2-27

Chapter 3: StorHouse database administration basics 3-1

People who manage StorHouse databases 3-1

Tools to help you manage StorHouse databases 3-2
StorHouse SQL statements 3-2
StorHouse Command Language 3-4

Resources to help you manage StorHouse databases 3-5
System parameters 3-5
User log 3-6
System tables 3-7

How to perform administration tasks 3-7
Submitting StorHouse SQL statements 3-8
Issuing StorHouse commands 3-8

Account for performing database and system administration tasks 3-9

Chapter 4: Accounts and privileges 4-1

About StorHouse accounts 4-1
Account IDs 4-2
Account passwords 4-2

About StorHouse privileges 4-3
Access privilege 4-3
Command privileges 4-4
SQLEXECUTE privilege 4-4
SQLCOMMAND privilege 4-4
Database privileges 4-4
DBA privilege 4-5
RESOURCE privilege 4-5
SCAN privilege 4-6
Database component privileges 4-6
Types of accounts and their privileges 4-7
SYSADM account 4-7
PUBLIC account 4-8
General database user accounts 4-8
Loader accounts 4-8
Unloader accounts 4-9
Metadata backup accounts 4-9
Local database application accounts 4-9
Host language application accounts 4-10
Ways to assign privileges 4-10
The WITH GRANT OPTION 4-11
When you grant to PUBLIC 4-12
System tables with privilege information 4-13
Summary of StorHouse privileges 4-13

Creating a StorHouse account 4-14

Granting database privileges 4-16

Granting database component privileges 4-17

Revoking database privileges 4-18

Revoking database component privileges 4-19

Changing a StorHouse account password 4-20

Adding an access or command privilege 4-21

Removing an access or command privilege 4-21

Listing StorHouse account IDs 4-22

Listing access and command privileges 4-23

Listing the privileges granted to PUBLIC 4-23

Listing database privileges 4-24

Listing database component privileges 4-25

Removing a StorHouse account 4-27

Chapter 5: User tablespaces 5-1

About user tablespaces 5-1
- Subspaces 5-2
- Storage specifications 5-4
  - Volume set 5-4
  - File set 5-5
  - Object type 5-5
  - Access Time Factor 5-7
  - Vulnerability Time Factor 5-8
  - Error Detection Code 5-9
  - File access group 5-10
  - Maximum data extent size 5-10
  - Extent retention in the performance buffer 5-11
- User tablespace examples 5-12
  - Example 1: Minimal volume mounts 5-13
    - Storage specifications 5-13
    - Tablespace assignments 5-13
  - Example 2: Simultaneous removal 5-14
    - Storage specifications. 5-14
    - Tablespace assignments 5-14
  - Example 3: Load concurrency 5-15

Storage specifications 5-15
Tablespace assignments 5-15
Example 4: Access concurrency 5-16
Storage specifications 5-16
Tablespace assignments 5-18
Subspace selection during data load, index load, and segment merge operations ..5-18
Default user tablespaces 5-19
Default user tablespace for a user table 5-19
Default user tablespace for a LOB column 5-19
Default user tablespace for an index 5-20
User tablespace naming conventions 5-20
User tablespace names 5-20
Tablespace IDs 5-21
Subspace numbers 5-21
System tables with user tablespace information 5-21
What's different about StorHouse user tablespaces? 5-22

Setting up a user tablespace 5-22
Creating volume sets and file sets 5-23
Creating a user tablespace 5-25

Assigning a default user tablespace 5-28

Listing accounts and their default user tablespaces 5-29

Changing a default user tablespace 5-29

Displaying default storage specifications 5-30

Altering a user tablespace 5-32

Listing user tablespace names and IDs 5-33

Displaying storage specifications for a user tablespace 5-34

Dropping a user tablespace 5-35

Chapter 6: User tables 6-1

About StorHouse user tables 6-1
Process of creating a user table 6-2
Owners and creators of user tables 6-3
User table naming conventions 6-3
User table names 6-3
Table IDs 6-4
System tables with user table information 6-4
What's different about StorHouse user tables? 6-5

Designing a column of a user table 6-5
Deciding the order of columns 6-6
Choosing a column name 6-6
Determining the data type 6-6
Using null values in a column 6-9
Specifying defaults for a column 6-10
Specifying LOB storage options 6-12
To set the maximum length for in-line LOB data 6-12
To store LOB values in a separate LOB subsegment file 6-13
To share storage with another LOB column 6-14
To store LOB values in a different user tablespace 6-14

Creating a user table 6-15

Displaying information about tables 6-18

Listing table names and IDs 6-19

Displaying a column definition 6-19

Displaying volume set, file set, and segment information 6-21

Dropping a user table 6-25

Chapter 7: Indexes for user tables 7-1

About StorHouse indexes 7-1
Value indexes 7-2

Queries that use value indexes 7-3
Value indexes and join processing 7-3
How value indexes are stored 7-4
Hash indexes 7-4
Queries that use hash indexes 7-4
How hash indexes are stored 7-5
Range indexes 7-5
Queries that use range indexes 7-6
Range indexes and extractor processing 7-7
How range indexes are stored 7-7
Simple and compound indexes 7-8
Default index type 7-9
Index owners 7-9
Index naming conventions 7-9
Index names 7-9
Index IDs 7-10
System tables with index information 7-10
What's different about StorHouse indexes? 7-10

Creating an index for a user table 7-11

Displaying information about indexes 7-12

Listing index names and IDs 7-14

Displaying volume set, file set, and segment information 7-15

Dropping an index 7-19

Chapter 8: Views 8-1

About views 8-1
How views work 8-1
Owners and creators of views 8-3
View naming conventions 8-4
View restrictions 8-4
View privileges 8-5
System tables with view information 8-5

Creating a view 8-6
Creating views from multiple tables 8-7
Naming columns in views 8-9

Displaying information about views 8-10

Listing view names 8-11

Dropping a view 8-12

Chapter 9: Synonyms 9-1

About synonyms 9-1
Private and public synonyms 9-2
Synonym owners 9-2
Synonym naming conventions 9-2
System table with synonym information 9-3

Creating a synonym 9-3

Renaming a synonym 9-4

Displaying information about synonyms 9-5

Listing synonym names 9-6

Dropping a synonym 9-7

Chapter 10: Metadata backup 10-1

About metadata backup 10-1
Which files are backed up? 10-2
When should you run a backup? 10-2
Who can run a backup? 10-3
What should you do before a backup? 10-3
What happens during a backup? 10-4
What should you do after a backup? 10-4
Where are metadata backup files stored? 10-5
How are backup files named? 10-6

How many backup files can you keep? 10-7

Invoking a metadata backup 10-8
Running a metadata backup as needed 10-9
Scheduling a metadata backup 10-10

Metadata backup error messages 10-11

Listing metadata backup files 10-16

Appendix A: StorHouse system tables A-1

About system tables A-1
System table names A-2
System table privileges A-3
List of system tables A-4

SYSCOLAUTH A-6

SYSCOLUMNS A-7

SYSDBAUTH A-8

SYSINDEXES A-10

SYSPACKAGE A-12

SYSPACKSTMT A-14

SYSRANGES A-16

SYSSMUSERS A-19

SYSSTAT_COL A-20

SYSSTAT_HIST A-21

SYSSTAT_IDX A-21

SYSSTAT_SMATRIX A-23

SYSSTHFILES A-24

SYSSTHSEGMENTS A-25

SYSSTHSPACES A-26
SYSSYNONYMS A-29
SYSTABAUTH A-30
SYSTABLES A-32
SYSTBLSPACES A-34
SYSVIEWS A-35

Appendix B: System limits B-1

Appendix C: System parameters for StorHouse/RM C-1

About system parameters C-1
System parameter types C-1
System parameter access C-2
System parameter list C-3
System parameter descriptions C-4
LOG_SQL_STMT C-4
LOG_SQL_TRANS C-4
SQL_BKUP_ACCOUNT C-5
SQL_BKUP_FSET C-5
SQL_BKUP_GROUP C-5
SQL_BKUP_LIMIT C-6
SQL_BKUP_VSET C-6
SQL_BLD_INDX_MEM C-6
SQL_HOLD_DATA C-7
SQL_HOLD_INDX C-7
SQL_HOLD_SPECIAL C-8
SQL_INDX_TYPE C-8
SQL_LDR_ENGINES C-9
SQL_LDR_MAXINTO C-9
SQL_LDR_MAXLOAD C-10
SQL_MAX_EXT_DATA C-10

SQL_MAX_EXT_HASH C-11
SQL_MAX_EXT_VAL C-11
SQL_SESSIONS C-12
STORHOUSE_REL C-12
SYSTEM_ID C-12

Displaying a system parameter value C-13

Setting or changing a system parameter value C-14

Index

FileTek

Welcome

The *StorHouse Database Administration Guide* describes how to create and manage the components of a StorHouse database.

StorHouse family of products

StorHouse® is the FileTek® enterprise-wide solution for managing the capture, storage, movement, and access of gigabytes to petabytes of relational and non-relational detail data. StorHouse technology combines industry-leading, scalable storage devices and Open System processors with specialized storage management and relational database management system (RDBMS) software components.

StorHouse/SM

StorHouse/SM, the storage management component, controls a hierarchy of storage devices comprised of cache, redundant array of independent disk (RAID), erasable and write-once-read-many (WORM) optical disk jukeboxes, and automated tape libraries. StorHouse/SM is also responsible for automating critical system management tasks, like data migration, backup, and recovery. StorHouse/SM comes standard with all StorHouse systems.

StorHouse/RM

StorHouse/RM, the RDBMS component, works in conjunction with StorHouse/SM to specifically administer the storage, access, and movement of relational data. StorHouse/RM provides row-level SQL access to high volumes of detail data on any layer—including tape—in the StorHouse storage hierarchy. SQL access is available from different platforms through a variety of industry-standard protocols. StorHouse/RM runs on Sun™ Solaris™ and Hewlett-Packard HP-UX platforms.

Control Center

StorHouse *Control Center* (CC) is the FileTek Windows®-based network computing system for providing administrative control of the StorHouse family of products. Control Center works with StorHouse/SM release 4.2 and higher and consists of one or more Control Center servers that communicate with Control Center clients over a TCP/IP network. The *Control Center server*, which runs on Windows NT, XP Pro, and 2000 platforms, provides network connectivity to StorHouse. The *Control Center clients*, which run on Windows 95, 98, 2000, XP Pro, and NT platforms, consist of one or more graphical user interface (GUI) modules for performing StorHouse system and database administration tasks, configuring and managing Control Center servers, and analyzing and monitoring StorHouse activity and performance.

Purpose of this document

The *StorHouse Database Administration Guide* describes how to manage StorHouse databases. Database administration tasks include creating database user tables and indexes, managing privileges for accessing database data, and assigning volume sets and file sets and other storage specifications to user tablespaces.

This manual also describes the StorHouse system administration tasks that are associated with StorHouse/RM. These tasks include managing the StorHouse signon accounts that access StorHouse databases, creating volume sets and files sets for database data, and setting various StorHouse system parameters for the efficient operation of StorHouse/RM.

This manual focuses on StorHouse database administration instead of system administration. It does not describe StorHouse hardware configurations, software features, storage allocation and control concepts, or StorHouse Command Language syntax. Refer to the manuals in the StorHouse User Document Set for more information about these topics.

This manual does not describe how to create databases. Contact FileTek Customer Support or your account system engineer for more information about creating databases. You can also create databases with the Control Center StorHouse/Admin module.

You can manage StorHouse databases with a command-based interface (SQL statements and StorHouse Command Language commands) or with a GUI interface (StorHouse/Admin). This manual shows the command-based interface. Refer to the Control Center User Document Set for more information about using the StorHouse/Admin GUI interface to manage StorHouse databases.

Intended audience

The *StorHouse Database Administration Guide* is intended for StorHouse database administrators (DBAs). A chief DBA typically oversees the operation and maintenance of all StorHouse databases. A departmental DBA typically manages one or more StorHouse databases. Depending on your organization, you or a separate StorHouse system administrator may manage StorHouse system tasks.

This manual assumes that you understand Structured Query Language (SQL) and StorHouse concepts. It also assumes that you have successfully completed these two FileTek training courses:

- StorHouse Overview and Command Language
- StorHouse System Administration

Contact your FileTek customer support representative for information about how to register for these courses if you have not already completed them. It is critical that you satisfy these prerequisites so that you can work effectively and knowledgeably with the StorHouse system administrator at your site.

Note: In this guide, *you* refers to both the StorHouse system administrator and to StorHouse DBAs.

Contents

This document is organized as follows:

- Chapter 1, “Introduction,” describes StorHouse/RM features and software components.
- Chapter 2, “StorHouse database concepts,” describes database components and naming conventions and how StorHouse stores and manages database data.
- Chapter 3, “StorHouse database administration basics,” describes the tools and resources available for managing StorHouse databases.
- Chapter 4, “Accounts and privileges,” describes StorHouse accounts and privileges that enable users to access StorHouse database data.
- Chapter 5, “User tablespaces,” defines user tablespaces and explains how to maintain them.

- Chapter 6, “User tables,” defines user tables and explains how to manage them.
- Chapter 7, “Indexes for user tables,” defines indexes and explains how to manage them.
- Chapter 8, “Views,” defines views and explains how to manage them.
- Chapter 9, “Synonyms,” defines synonyms and explains how to manage them.
- Chapter 10, “Metadata backup,” explains how to back up the metadata and range indexes for one or multiple databases.
- Appendix A, “StorHouse system tables,” describes the system tables used by StorHouse/RM.
- Appendix B, “System limits,” lists the maximum limits for various components of StorHouse/RM.
- Appendix C, “System parameters for StorHouse/RM,” describes the StorHouse system parameters that are used to configure StorHouse/RM.

Related documentation

The following FileTek publications can help you use and understand StorHouse/RM.

Other StorHouse/RM publications

In addition to the *StorHouse Database Administration Guide*, the StorHouse/RM User Document Set consists of the following publications:

- The *FileTek MVS Data Loader Utility Manual*, publication number 900109, describes how to load data into StorHouse user tables from an MVS environment.
- The *FileTek FTP Data Loader Manual*, publication number 900115, describes how to load data into StorHouse user tables from a UNIX, VAX, or other host environment with File Transfer Protocol (FTP).
- The *FileTek FTP Data Unloader Manual*, publication number 900137, explains how unload data from StorHouse user tables with FTP.
- The *StorHouse SQL Reference Manual*, publication number 900111, describes the SQL statements, predicates, and functions supported by StorHouse.
- The *StorHouse SQL Quick Reference*, publication number 900122, contains a summary of the material in the *StorHouse SQL Reference Manual*.
- The *StorHouse ESQL Manual*, publication number 900121, explains how application programs can use the StorHouse ESQL interface to submit SQL statements to retrieve data in StorHouse user tables.
- The *StorHouse/RM Glossary*, publication number 900112, defines terms used in the StorHouse/RM User Document Set.
- The *StorHouse/RM Metadata Conversion Manual*, publication number 900142, explains how to convert metadata from one StorHouse/RM release to another.
- The *StorHouse/UDB Link Installation and Configuration Manual*, publication number 900163, explains how to install the StorHouse/UDB Link and how to configure IBM® DB2® Universal Database (DB2 UDB) to work with

StorHouse. It also discusses security management, data modeling, and DB2 components that support federation.

StorHouse publications

Some of the StorHouse publications that can help you understand StorHouse/RM are:

- The *StorHouse Glossary*, publication number 900027, defines the terminology used in FileTek StorHouse publications.
- The *StorHouse Concepts and Facilities Manual*, publication number 900026, defines the basic concepts, structures, and functions of StorHouse.
- The *Command Language Reference Manual*, publication number 900005, describes StorHouse Command Language commands, system parameters, and command privileges.
- The *System Administrator's Guide*, publication number 900007, describes system recovery, user file backup, account administration, and storage management procedures and concepts for StorHouse.
- The *Messages and Codes Manual*, publication number 900032, lists all StorHouse system and host software messages.

Control Center publications

The following Control Center publications describe how to use StorHouse/Admin to complete StorHouse system and database administration tasks.

- *Getting Started with StorHouse/Admin*, publication number 900135, contains an introduction to the software features, windows, menus, tool bars, and selected system and database management tasks.

- The *StorHouse/Admin System Administrator's Quick Reference*, publication number 900147, contains procedures for performing system administration tasks with StorHouse/Admin. Some tasks include setting system parameters, creating file access groups, changing volume attributes, and printing labels.

- The *StorHouse/Admin Database Administrator's Quick Reference*, publication number 9001487, contains procedures for performing database administration tasks with StorHouse/Admin. Some tasks include creating databases, granting and revoking privileges, using ISQL, archiving and backing up segments, and backing up and restoring metadata.

- The *StorHouse/Admin System Operator's Reference*, publication number 900149, contains procedures for performing system operator tasks with StorHouse/Admin. Some tasks include bringing up a StorHouse device, reserving the system, monitoring user activity, and shutting down the system.

Conventions

This book uses the following conventions for illustrating statement and command formats, presenting examples, and identifying special terms:

Convention	Meaning
UPPERCASE HELVETICA	SQL statement and StorHouse command examples
Italics	New terms, emphasized text, and publication titles
▼	Procedures

Chapter

1

Introduction

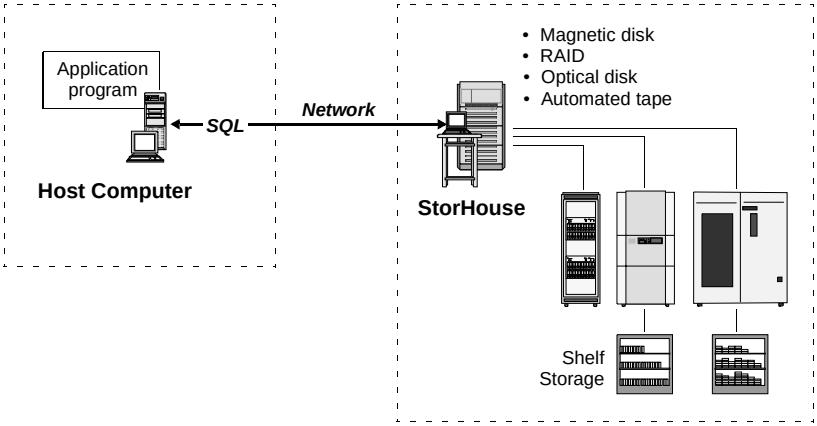
This chapter provides an overview of StorHouse/RM. It describes the basic functions, features, and software components of StorHouse/RM.

What is StorHouse/RM?

StorHouse/RM is specialized relational database management software for managing the storage, access, and movement of relational data across the enterprise. StorHouse/RM works in conjunction with StorHouse/SM to provide a storage repository for high volumes of detail data.

With StorHouse/RM, you load database data from your host computer into StorHouse database tables using FileTek data loading processes. Once data is loaded into StorHouse, client applications and database tools running on a variety of host platforms can access selected rows of information on any layer in the StorHouse storage hierarchy using industry-standard Structured Query

Language (SQL). The following graphic shows an example of how StorHouse/RM provides SQL access to the StorHouse storage hierarchy.



StorHouse/RM features

StorHouse/RM provides the features you would expect of an advanced RDBMS, as well as other features that uniquely suit it to storing and retrieving vast amounts of data on StorHouse.

Standard SQL. The SQL you use to access StorHouse databases is a subset of the American National Standards Institute (ANSI) SQL-99 standard plus extensions defined by FileTek to support additional capabilities.

System managed storage. Database tables and indexes are stored on an automatically managed hierarchy of media and devices including magnetic disk, Redundant Array of Independent Disks (RAID), optical disk, automated tape, and shelf storage.

Remote database interfaces. Database gateways enable applications based on an existing RDBMS, such as IBM DB2 and Oracle®, to interface with StorHouse.

In this case, StorHouse is a *remote database* and the RDBMS is a *local database*. Gateways with network communications support are a class of software called *middleware*.

Federation. StorHouse can serve as a data source in a federated system. The *StorHouse/UDB Link* is the FileTek software that implements the connection between DB2 UDB and StorHouse databases. This software gives DB2 UDB users near-transparent access to the terabytes to petabytes of data supported by StorHouse/RM.

Application program interface. StorHouse SQL provides an *Embedded SQL (ESQL) interface* that supports coding SQL statements in C and C++ programs. By embedding SQL statements in a host program, you can develop applications that are more flexible than those developed in just the host language or SQL.

Bulk data loading and unloading. Comprehensive data loader programs developed by FileTek transfer large amounts of relational data from host environments to StorHouse. The FileTek data unloader utility copies data from StorHouse database tables to a local file on a client system.

Large object (LOB) support. StorHouse/RM supports the storage and access of Binary Large Objects (BLOBs) and Character Large Objects (CLOBs). BLOBs include data such as video, photos, and voice up to 2 gigabytes (GB) in size. CLOBs include large character-based data up to 2 GB in size.

Concurrency. The StorHouse/RM architecture facilitates concurrent access by multiple users of local databases, multiple application programs, and multiple loading and unloading processes.

Indexing. Comprehensive database indexing speeds access to data in large complex tables. StorHouse supports three index types: hash, value, and range.

High-speed extraction. The StorHouse extractor provides a practical solution for retrieving massive amounts of database data. It is a performance enhancement

feature for queries that result in a *full table scan* (a search in every row in a database table without using an index) or a *full segment select* (a query that returns data from one or more entire segments with the use of a range index).

Database integrity. A reliable system of constraints, logging, recovery, and metadata backup ensure database integrity.

Controlled access/database security. A complete set of privileges controls user access to StorHouse databases and database data.

StorHouse/RM software components

The major StorHouse/RM software components are:

- Data loaders
- Data unloader
- StorHouse engines

Data loaders

FileTek uses bulk data loading processes to load table data into StorHouse. You can load data from an MVS environment with the FileTek MVS Data Loader utility, and you can load data from an FTP-enabled host with the FileTek FTP Data Loader.

FileTek MVS Data Loader utility

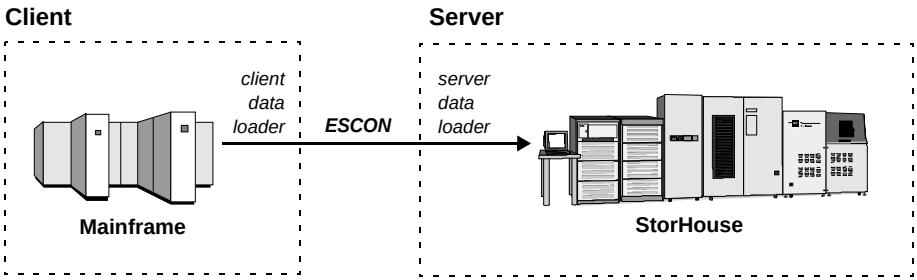
The *FileTek MVS Data Loader utility* is an MVS batch program that initiates the loading of a sequential dataset from a host computer into a StorHouse user table.

This process requires cooperation between two separate FileTek data loader programs:

- The *client data loader*, which runs on your host computer, prepares your data for loading and sends it to StorHouse. The FileTek MVS Data Loader utility is the FileTek-supplied client data loader.

- The *server data loader*, which runs on StorHouse, loads your data into StorHouse user tables and builds and stores any indexes.

The FileTek MVS Data Loader utility bypasses the gateways when transferring data from the host to the server, relying instead on an existing channel connection and the StorHouse application program interface (API) to achieve maximum data transfer rates.

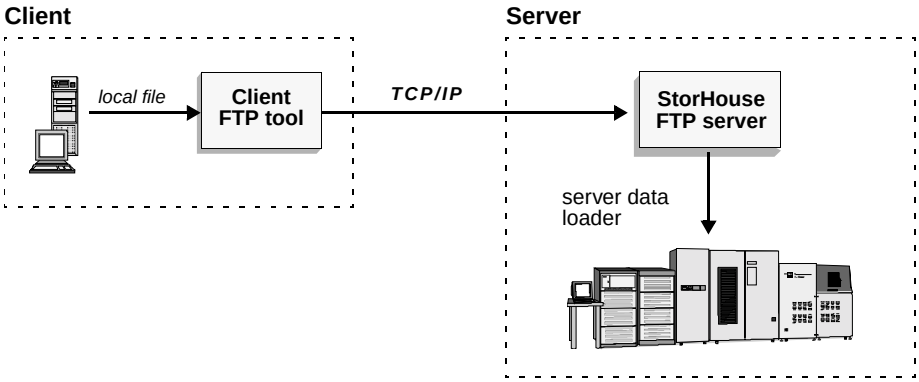


Refer to the *FileTek MVS Data Loader Utility Manual* for information about loading data from an MVS host.

FileTek FTP Data Loader

The *FileTek FTP Data Loader* is a tool for loading data into StorHouse user tables from a host that's enabled with FTP. With the FileTek FTP Data Loader, you use your standard client FTP client software to communicate with the FileTek StorHouse FTP server. These two programs communicate over a TCP/IP

connection to transfer files from your local file system on your host to a remote file system on StorHouse.



Refer to the *FileTek FTP Data Loader Manual* for information about loading data via FTP.

Data unloader

The FileTek FTP Data Unloader is a tool for copying data from StorHouse database tables to your client system. It executes a SELECT statement, then formats and transfers the result data according to your specifications. With the FileTek FTP Data Unloader, you use your standard FTP client software to communicate with the StorHouse FTP server. Some of the features of the FileTek FTP Data Unloader include the following:

- You can unload an entire table, specific columns or rows of a table, or multiple tables (join).
- Depending on your host environment, you can pipe the output to another program, such as a data loading utility.

- You can receive the result data in one of three record formats: text, fixed-length, or variable length.
- You can receive the result data in a sequential (or flat) file on your computer or in a Virtual Record Access Manager (VRAM™) file on StorHouse.
- You can receive LOB data in separate result files (one LOB value per file) on your client computer or on a remote system.

Refer to the *FileTek FTP Data Unloader Manual* for information about unloading data via FTP.

StorHouse engines

A StorHouse *engine* is the main StorHouse/RM process. The main functions of a StorHouse engine are as follows.

Parsing and optimizing SQL. A StorHouse engine checks your privileges, parses, then optimizes an SQL statement before executing it. The *parser* analyzes the request for syntax and semantics, then breaks it into basic components for the optimizer. The *optimizer* develops an access plan that most efficiently completes the query.

Retrieving data from StorHouse. When you submit an SQL SELECT statement, StorHouse/RM assigns an engine to execute the query. The StorHouse engine parses and optimizes the request, identifies the table and rows within the table that it wants to access, retrieves the information, and forwards it to the requesting host.

Assisting a data loader. During a load, a StorHouse engine checks your StorHouse privileges and obtains the metadata to store the table, index, and LOB data. When a load completes, a StorHouse engine updates the metadata. One StorHouse engine is assigned to each load.

Chapter

2

StorHouse database concepts

This chapter contains an overview of StorHouse databases. It explains:

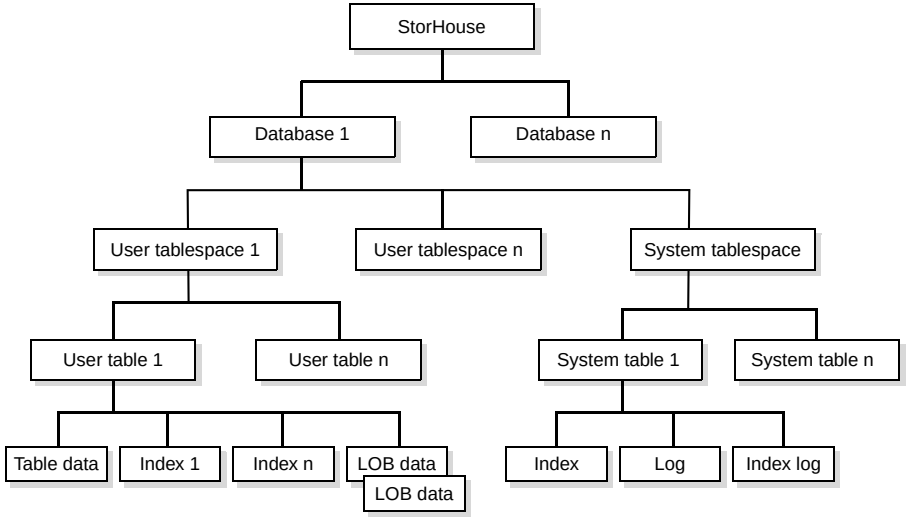
- Types of database components, both system and user
- Segments and their relationship to table data, indexes, and LOB data
- Conventions for naming databases, database components, and segments

StorHouse database components

A *StorHouse database* is a collection of user and system components used for storing and accessing read-only, detail data. In general, database user components are the database information that you store, access, and view while database system components describe a database. User components reside on and are managed by the StorHouse/SM file system. System components reside on and are managed by the UNIX file system.

The following graphic illustrates the user and system components of a StorHouse database. It shows all user table components assigned to the same user tablespace.

You can also assign table data to one user tablespace and indexes and LOB data to different user tablespaces, as shown in the graphic on page 2-3.



StorHouse provides both a logical and a physical structure to system and user data. The logical structure (data representation) is accomplished with tables. The physical structure (data storage) is accomplished with files. Tables represent data, and files contain data. Users view and access data as relational tables. StorHouse/RM accesses data stored in files.

Database user components

Database user components are structures you work with to manage the storage and retrieval of data. StorHouse database user components are:

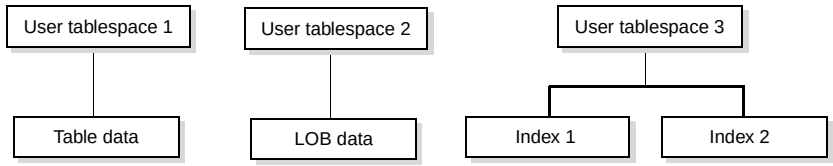
- User tablespaces
- User tables
- User table indexes
- Views
- Synonyms

User tablespaces

A *user tablespace* is a logical structure that defines how to store table, index, and LOB data on StorHouse. Before creating and loading user tables, you create one or more user tablespaces. Each user tablespace consists of one or more *subspaces* that define the storage specifications for table data, hash index entries, value index entries, and LOB data.

When you create a user table, you assign it to a user tablespace. The table data then belongs to that user tablespace. If the user table contains a LOB column, you can assign that LOB column to a different user tablespace or use the default, which is the same user tablespace as the table. And when you create an index for the user table, you can assign that index to the same user tablespace as the table or to a different one. The components then of a user table can belong to multiple user tablespaces, as shown in the following graphic.

Multiple user tablespaces for a user table's components



If you don't explicitly assign user tables, LOB columns, and indexes to a user tablespace, StorHouse/RM attempts to assign them to a default tablespace.

See Chapter 5, "User tablespaces," for more information about user tablespaces, subspaces, and storage specifications.

User tables

A *user table* holds user-accessible data. Logically, a user table consists of columns, rows, and data values. A row is also called a *tuple*. A column is also called a *field*. You define the columns of a user table with a CREATE TABLE statement. Then

you load data into that user table with a FileTek data loader. *User table components* are all of the data—table data, index entries, LOB data—that refer to the same user table.

Physically, user tables are stored in one or more *segments*. When you load data, StorHouse/RM saves the table data as a table data file in a segment. See "Segmentation" on page 2-7 for more information about how table data is stored on StorHouse.

See Chapter 6, "User tables," for more information about user tables.

User table indexes

A *user table index* provides quick and efficient access to table data. You can create an index on a column (*simple index*) or combination of columns (*compound index*) in a user table. StorHouse supports three index types: range, value, and hash. When you create an index, you indicate the index type.

Physically, hash indexes and value indexes are stored in one or more index files in segments, and range indexes are stored in system tables. See "Segmentation" on page 2-7 for more information about how hash and value indexes are stored. See "SYSRANGES" on page A-16 for a description of the range index system tables.

See Chapter 7, "Indexes for user tables," for an explanation of index types, when StorHouse/RM uses different indexes, and how to create them.

Views

A *view* is a virtual table that appears and acts like a table but draws its contents from one or more existing tables and/or views. When you query a view, you actually query the tables referenced by the view.

For instance, you might construct a two-column view that provides a column of customer names drawn from one table and a column of associated sales representatives from another table. Or you might construct a view from an employee record table that excludes the column containing salary information.

Views are useful for tailoring or limiting user access to data. They provide convenience, security, or both by letting you determine which data in which tables is available to which users.

See Chapter 8, “Views,” for more information about working with views.

Synonyms

A *synonym* is another name that you can assign to a table, view, or synonym. You can create a *private synonym* for your own use or a *public synonym* for all accounts who can access the database. You use synonyms in place of the original table, view, or synonym name in SQL statements,

See Chapter 9, “Synonyms,” for more information about working with synonyms.

Database system components

Database system components are structures that StorHouse/RM uses to manage a database. These system components—also called *metadata*—reside on the UNIX file system. Like user components, system components are represented as tables and indexes and stored as files. StorHouse database system components are:

- System tablespace
- System tables
- System table indexes
- System table logs
- System table index logs

System tablespace

A *system tablespace* is a logical structure that contains the system components for a database. Each database has a system tablespace. On UNIX, a system tablespace is a directory, also called *database directory*. For each new database, StorHouse/RM

creates a database directory and a set of metadata under the default path /filetek/sth/sthdbs.

In the following graphic, for example, the CALLS.dbs directory in the UNIX file system contains the system component files for the CALLS database.

```
/filetek/sth/sthdbs/CALLS.dbs

CONFIG.CGF        I0000000011.IDX  T0000000003.TBL  T0000000015.LOG
I0000000000.IDX   I0000000011.LOG  T0000000004.LOG  T0000000015.TBL
I0000000000.LOG   I0000000012.IDX  T0000000004.TBL  T0000000016.LOG
I0000000001.IDX   I0000000012.LOG  T0000000005.LOG  T0000000016.TBL
I0000000001.LOG   I0000000013.IDX  T0000000005.TBL  T0000000017.LOG
I0000000002.IDX   I0000000013.LOG  T0000000006.LOG  T0000000017.TBL
I0000000002.LOG   I0000000014.IDX  T0000000006.TBL  T0000000018.LOG
I0000000003.IDX   I0000000014.LOG  T0000000007.LOG  T0000000018.TBL
I0000000003.LOG   I0000000015.IDX  T0000000007.TBL  T0000000019.LOG
I0000000004.IDX   I0000000015.LOG  T0000000008.LOG  T0000000019.TBL
I0000000004.LOG   I0000000016.IDX  T0000000008.TBL  T0000000020.LOG
I0000000005.IDX   I0000000016.LOG  T0000000009.LOG  T0000000020.TBL
I0000000005.LOG   I0000000017.IDX  T0000000009.TBL  T0000000021.LOG
I0000000006.IDX   I0000000017.LOG  T0000000010.LOG  T0000000021.TBL
I0000000006.LOG   I0000000018.IDX  T0000000010.TBL  T0000000022.LOG
I0000000007.IDX   I0000000018.LOG  T0000000011.LOG  T0000000022.TBL
...               ...              ...              ...
```

See page 2-26 for more information about UNIX file names for database system components.

System tables

A *system table* contains information about the contents of a database as well as account access to database components. System tables also contain range index entries. For each new database, StorHouse/RM creates a set of system tables and stores them as UNIX files in the system tablespace. StorHouse/RM then maintains all system tables except the SYSSMUSERS system table. Only accounts with the appropriate privileges should use SQL to access system tables.

See Appendix A,"StorHouse system tables," for descriptions of all system tables. Chapter 10 explains how to back up system tables and other database system components.

System table indexes

A *system table index* helps StorHouse/RM access information in a system table. Some, but not all, system tables have an index. When you create a database, StorHouse/RM creates the system table indexes and stores them as UNIX files in the system tablespace.

System table logs

Each system table has a corresponding *system table log* that's used to recover changes to system tables. Before StorHouse/RM updates a system table, it first copies a "before image" of any record being updated to the system table log and then makes the change in the system table. If the transaction fails or is rolled back, StorHouse/RM copies the before image (or *undo record*) back to the system table, removing the change. If the transaction completes or is committed, StorHouse/RM empties the system table log.

System table index logs

Each system table index has a log that's used to recover changes to system table indexes. The operation of the index log is the same as the system table log.

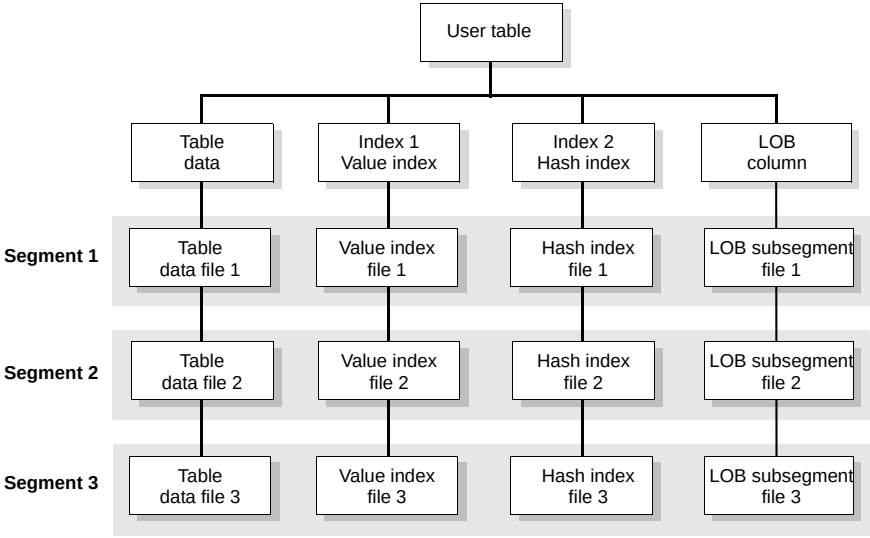
Segmentation

StorHouse user tables consist of one or more segments. A *segment* is a set of StorHouse files containing table data, index entries, and LOB data. Each time you load data into a user table, StorHouse creates a segment with one file for the table data, one file for each hash index, one file for each value index, and one or more subsegment files for each LOB column. The term *segment files* refers to the

table data files, hash index files, value index files, and LOB subsegment files that make up StorHouse segments.

Note: Depending on the actual size or by user request, LOB values may be stored in the table data file, or LOB values in different columns may be stored in the same LOB subsegment file, or LOB values in the same column may be stored in multiple LOB subsegment files.

The following user table consists of three segments. This user table has one value index, one hash index, and one LOB column.



Whether a user table is composed of one or multiple segments depends on the following:

- Size of the table data – The maximum size of a table data file is approximately 100 GB. If input data for a load exceeds 100 GB, you must load that data into multiple segments.

- Subsequent loads – When you load more data into a table, you create at least one (depending on the size of the data) new segment for each load.

- User request – You have the option of creating multiple segments during a load. For example, if you're loading data consisting of a year of transactions, you can create one segment for the January transactions, another segment for the February transactions, and so on.

Table data files

The data of a table is stored in a *table data file*. Each segment contains one table data file. The maximum size of a row in a table data file is 32,705 bytes. The maximum size of a table data file is approximately 100 GB. Table data files may contain:

- LOB values that fit within the row
- Object identifiers (OIDs) for LOB values that do not fit within the row

See "LOB storage and subsegment files" on page 2-10 for more information about object identifiers and ways to store LOB data.

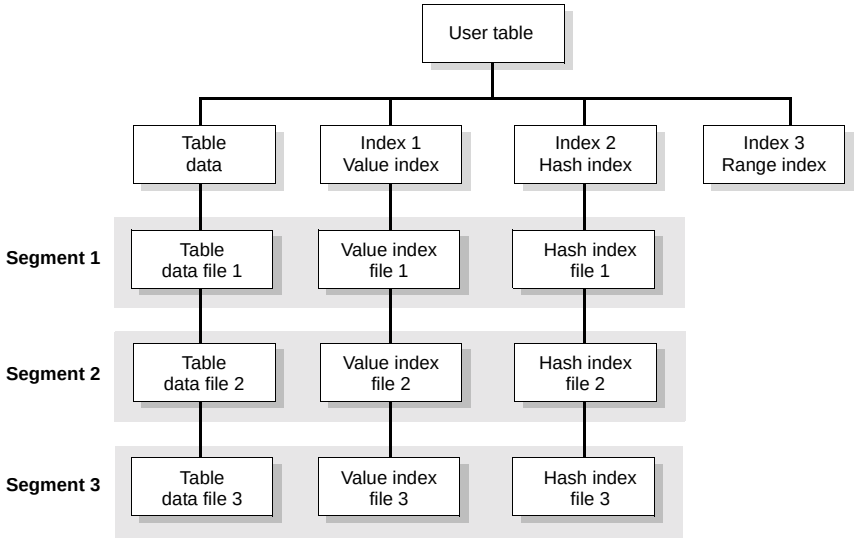
Index files

During a load, a FileTek data loader builds and stores any index entries for value, hash, and range indexes. The data loader stores:

- Value index entries in a *value index file*
- Hash index entries in a *hash index file*
- Range index entries in a set of system tables

Each segment contains one value index file for each value index on a table and one hash index file for each hash index on a table. A range index applies to all segments of a user table. When you load more data into a user table, a FileTek data loader creates a new segment with index files and inserts the range index entries into the system tables.

For instance, if you had created one value, hash, and range index for a user table, and you loaded data in January, February, and March, then after the March load, the user table would consist of three segments. Each segment would contain a table data file, a value index file, and a hash index file. The range index would contain entries for the three segments.



LOB storage and subsegment files

A user table can consist of multiple LOB columns defined as BLOB and CLOB data types. The maximum size of a LOB column is 2 GB. When creating a table, you can choose to store LOB values:

- In the table data file (space permitting)
- In a separate file, called a *LOB subsegment file*, with other LOB values in the same column
- In a separate LOB subsegment file with other values in other LOB columns

An *in-line LOB* is a LOB value that's stored with the table data. An *out-of-line LOB* is a LOB value that's stored in a separate file from the table data. StorHouse/RM determines in-line and out-of-line storage on a row-by-row basis.

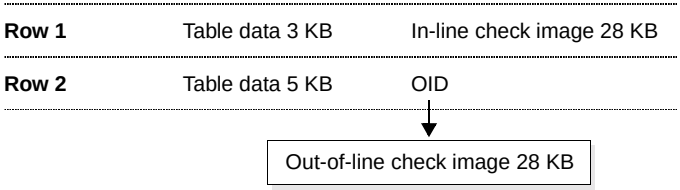
In-line LOBs

If space permits—a LOB value does not exceed any user-defined in-line limit and the row does not exceed 32 KB—you can store a LOB value in a table data file. StorHouse/RM treats in-line LOB values like other variable-length, nullable fields and determines in-line storage by the actual size of the LOB value, not by the maximum size defined for the LOB column. In-line storage is the default, that is, unless you specify otherwise, a FileTek data loader attempts to store LOB values with the table data.

You can set the maximum length for in-line LOB data, for instance, 15 KB. If you omit the length, the default is 32 KB. A FileTek data loader stores a LOB value in a row as long as the value does not exceed the maximum in-line length (like 15 KB) or the maximum row length (32 KB). If a LOB value exceeds either length, the data loader stores the value in a separate LOB subsegment file. For instance, if other columns in a table are 20 KB and a LOB value is 13 KB (within the 15 KB limit), the data loader stores the LOB value out-of-line because the total row size (in this example, 33 KB) exceeds the maximum (32 KB).

For example, assume you created a table for bank account information with check images that require approximately 28 KB of storage. You defined the check image column as BLOB data type and intend to store the images in-line with the table data when possible. Row 1 contains 31 KB of data and row 2 contains 33 KB. The FileTek data loader stores all table data and the row 1 check image in the

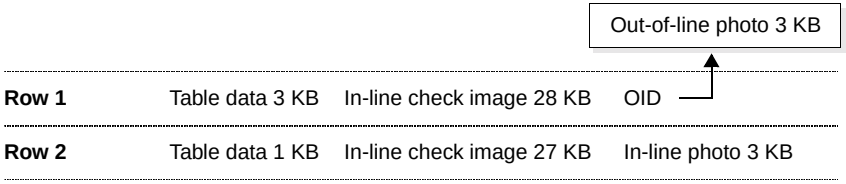
table data file and stores the row 2 check image in a separate LOB subsegment file. Row 2 contains an OID that identifies the LOB subsegment file.



Note: In the preceding example, the size of row 2 is 5 KB plus 22 bytes for the OID.

If a user table contains multiple LOB columns, a FileTek data loader determines in-line storage in column order. In other words, the data loader attempts to store the first LOB column value with the table data, and if space permits, attempts to store the second LOB column value with the table data, and so on.

For instance, if you defined a check image column before a photo ID column, the data loader first attempts to store the check image with the table data before the photo ID. In the following example, the data loader stores the row 1 check image in-line with the table data and the row 1 photo ID out-of-line in a LOB subsegment file. For row 2, the data loader stores both the check image and the photo ID in-line with the table data because both LOBs fit within the row.



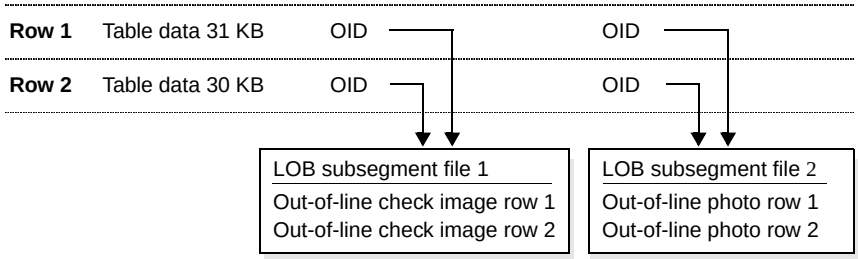
Note: In the preceding example, the size of row 1 is 31 KB plus 22 bytes for the OID.

Out-of-line LOBs

Out-of-line LOBs are stored in LOB subsegment files. For out-of-line LOBs, a FileTek data loader generates and stores *object identifiers* (OIDs) in rows in the table data file. StorHouse/RM uses OIDs to locate out-of-line LOB values. An OID, 22 bytes in size, contains the following:

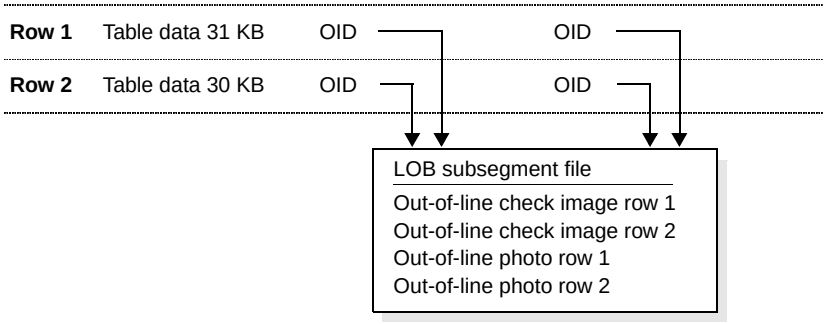
- LOB length (in bytes)
- LOB subsegment ID
- Starting frame number within the LOB subsegment file
- Offset within the LOB subsegment file

When defining a LOB column, you can choose to always store LOB values out-of-line or you can use the default—store LOB values in-line when possible and out-of-line when necessary. By default, a FileTek data loader attempts to store a column's values in the same LOB subsegment file. For instance, in the following example, the check images would reside in one LOB subsegment file and the photo IDs in a different LOB subsegment file.

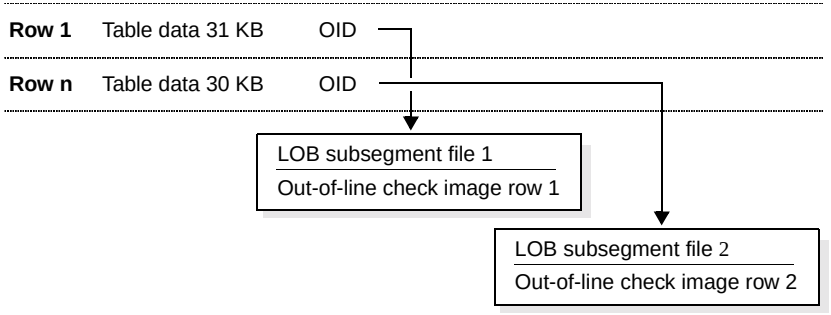


You can also choose to store values of different LOB columns in the same LOB subsegment file. In other words, multiple LOB values in multiple columns can

share the same LOB subsegment file. For instance, the check images and photo IDs can share one LOB subsegment file.

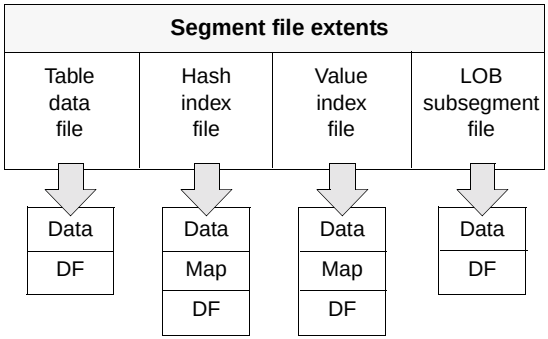


If a LOB subsegment file reaches the maximum file size (approximately 100 GB), a FileTek data loader creates a new file as needed. This means that values for a LOB column may actually be stored in multiple LOB subsegment files.



Extents

StorHouse files consist of *extents*, which are contiguous data or control information that StorHouse treats as a unit. The following graphic shows the different types of extents for segment files.



All files have data and DF extents. Index files also have map extents. Definitions of these extents are as follows:

- A *data extent* contains user data and/or control data.
- A *definitions (DF) extent* contains information necessary to retrieve the data.
- A *map extent* is the high-level index that StorHouse always reads first when doing index lookups.

A user tablespace defines the maximum size of extents and extent retention in the performance buffer. See “Maximum data extent size” on page 5-10 and “Extent retention in the performance buffer” on page 5-11 for more information about managing extents.

Storage management on StorHouse

StorHouse/SM provides a file type, called STORHOUSE, for segment files. The StorHouse/SM software automatically manages the movement of these files

through the storage hierarchy based on the storage specifications in a user tablespace. You can also:

- Back up and/or archive files to create duplicate copies for recovery
- Recover files should they become unreadable
- Invalidate segments in a user table, making the files inaccessible
- Merge segments, for instance, consolidate a group of small segments into one segment

Refer to the StorHouse *System Administrator's Guide* for more information about backing up, archiving, and recovering files. Refer to the *FileTek FTP Data Loader Manual* or the *FileTek MVS Data Loader Utility Manual* for more information about invalidating and merging segments.

Naming conventions

This section describes naming conventions for StorHouse databases and database components. In general:

- Database user components have user-assigned names and system-assigned identifiers.

- Segments have user-assigned tags and system-assigned identifiers and StorHouse file names.

- Database system components have system-assigned identifiers and UNIX file names.

Database names

A StorHouse database name:

- Must be unique
- Must start with a letter
- Cannot exceed 32 characters
- Is case sensitive
- Can consist of any combination of a–z (lowercase), A–Z (uppercase), 0–9, and underscore (_)

Some considerations for naming databases are as follows:

- If you use StorHouse in conjunction with a local database, follow the naming rules for your local database. For example, if you use DB2 and DRDA, a StorHouse database name must be uppercase and cannot exceed 16 characters.

- StorHouse uses the database name (stored in uppercase) as part of the file name for segment files. For database names longer than 16 characters, StorHouse uses a database alias instead of the actual database name. See "Segment and LOB subsegment identifiers" on page 2-20 for more information about database aliases.

- If you plan to export database data from one StorHouse system and import that data into another StorHouse system, the database name cannot be longer than 16 characters.

- You cannot use case to differentiate StorHouse database names. For instance, you can't name one database CUSTOMER and a second database customer or Customer. Database names must be unique.

Database user component names

When you create a database user component, you give it a name. You assign names to user tablespaces, user tables, columns, user table indexes, views, and synonyms. A database user component name should follow these SQL identifier conventions:

- Start with a letter
- Cannot be a StorHouse SQL reserved word (refer to the *StorHouse SQL Reference Manual* for a list of reserved words)
- Can contain up to 32 contiguous characters (no blanks) including a-z (lowercase), A-Z (uppercase), 0-9, and underscore (_)

Delimited names

If a database user component name does not follow SQL identifier conventions, you must delimit it with double quotes in SQL statements. You must also use the appropriate case because delimited SQL identifiers are case sensitive.

For example, if you create a user tablespace with this delimited name:

```
CREATE TABLE SPACE “oct1999”
```

then you must delimit the name and use the same case in other SQL statements:

```
ALTER TABLE SPACE “oct1999”
```

Unique names

Database user component names must be unique as follows:

- User tablespace names must be unique within a database.
- For user tables, the combination of owner and table name must be unique from all table, view, and synonym names in a database.
- Column names must be unique within a user table, but they do not have to be unique within a database.
- For indexes, the the combination of owner and index name must be unique from all other index names in a database.
- For views, the combination of owner and view name must be unique from all table, view, and synonym names in a database.
- For synonyms, the combination of owner and synonym name must be unique from all table, view, and synonym names in a database.

Database user component identifiers

StorHouse/RM assigns numeric component identifiers (IDs) to user tablespaces, user tables, and indexes. You can use these IDs to access information in system tables. The following rules apply to system-assigned identifiers.

- Tablespace IDs – Tablespaces are numbered sequentially within a database. User tablespace IDs start with 2.
- Table IDs – Tables are numbered sequentially within a database, meaning no two tables have the same component ID. User table IDs start with 100.

- Index IDs – Indexes are numbered sequentially within the database, meaning no two indexes have the same component ID. User table index IDs start with 100.

Segment and LOB subsegment identifiers

StorHouse/RM assigns each segment a *segment ID*, numbered sequentially by user table. The first segment ID for each user table is 0, the second segment ID is 1, and so on.

StorHouse/RM assigns each LOB subsegment file a *subsegment ID*, numbered sequentially within a segment. The first subsegment ID is 0. If a user table contains one LOB column, and that LOB column requires multiple subsegment files, then subsegment IDs increment sequentially, that is, the first subsegment ID is 0, the second subsegment ID is 1, and so on.

If a user table contains multiple LOB columns, and each LOB column requires one LOB subsegment file, then subsegment ID 0 identifies the first LOB column, subsegment ID 1 identifies the second LOB column, and so on.

If a user table contains multiple LOB columns (for instance, two) and each LOB column requires multiple LOB subsegment files (for instance, two), then subsegment ID 0 identifies the first file for the first LOB column, subsegment ID 1 identifies the first file for the second LOB column, subsegment ID 2 identifies the second file for the first LOB column, and subsegment ID 3 identifies the second file for the second LOB column.

Segment tags

During a load, you can optionally name the current segment you're loading in case you need to replace or merge the segment in the future. This name is called a *segment tag*. If you don't supply a segment tag, the load ID is the default. Segment tags are stored in the SYSSTHSEGMENTS system table. Refer to the

FileTek MVS Data Loader Utility Manual or the *FileTek FTP Data Loader Manual* for more information about naming, replacing, and merging segments.

Segment file names

Segment files have StorHouse file names. You can use file names in StorHouse Command Language commands. The file naming convention for these files may differ by StorHouse/RM release. A file always retains the original file name.

StorHouse/RM 3.2 segment file names

The file naming convention for table data files, index files, and LOB subsegment files created in StorHouse/RM 3.2 is as follows:

database_name.system_id.typecomponent_id.segment_id.lob_subsegment_id

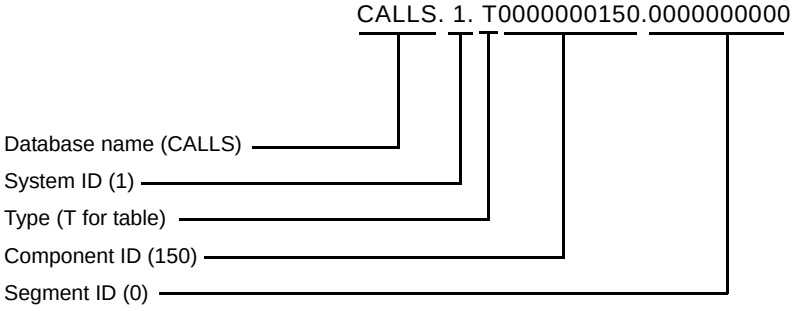
File naming convention for StorHouse/RM 3.2

File name item	Length	Description
database_name	up to 16 characters	Name of the StorHouse database. If the database name exceeds 16 characters, StorHouse uses a database alias consisting of the first 12 characters of the database name, followed by the @ character and three decimal digits. For example, StorHouse replaces the database name BILLDETAILWAREHOUSE with the database alias BILLDETAILWA@001.
system_id	up to 6 decimal digits	Value of the StorHouse SYSTEM_ID system parameter
type	1 character	Type of component: ■ H for hash index ■ L for LOB ■ T for table ■ V for value index

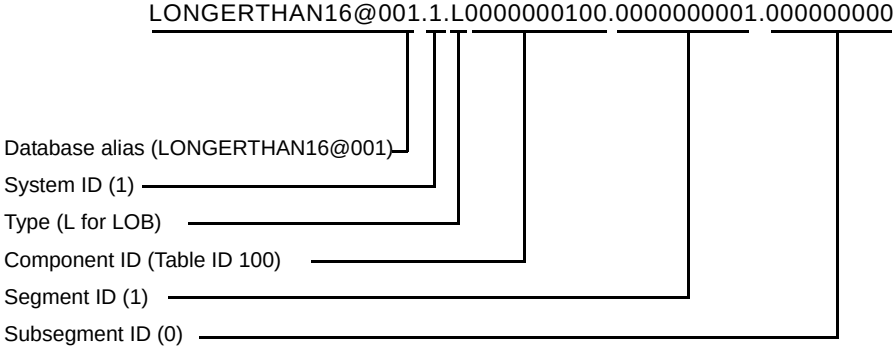
File naming convention for StorHouse/RM 3.2 (continued)

File name item	Length	Description
component_id	10 decimal digits	Table ID or index ID. If the type is L (for LOB), the component_id is the table ID.
segment_id	10 decimal digits	Segment ID containing the file
lob_subsegment_id	9 decimal digits	Subsegment ID containing the LOB

For example, the following StorHouse file name identifies a table data file in the first segment (segment ID 0) of a user table (table ID150) in the CALLS database.



The following StorHouse file name identifies the first LOB subsegment file in the second segment (segment ID1) of a user table (table ID 100) in the LONGERTHAN16CHARACTERSDBNAME database.



StorHouse/RM 2.3 segment file names

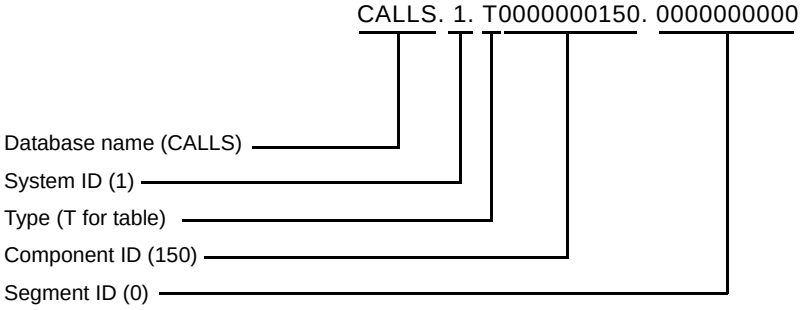
The file naming convention for table data and index files created in StorHouse/RM 2.3 is as follows:

database_name.system_id.typecomponent_id.segment_id

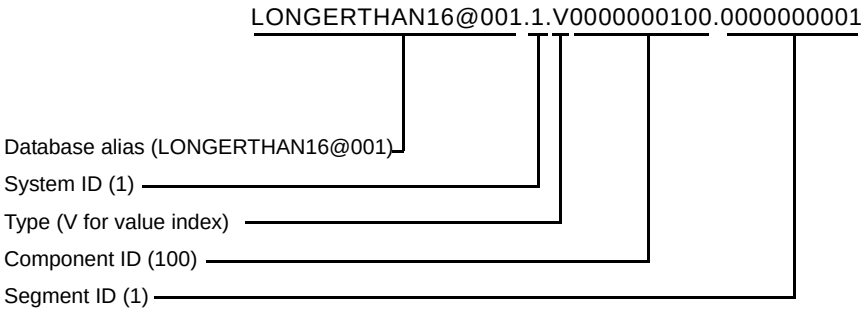
File naming convention for StorHouse/RM 2.3

File name item	Length	Description
database_name	up to 16 characters	Name of the StorHouse database. If the database name exceeds 16 characters, StorHouse uses a database alias.
system_id	up to 6 decimal digits	Value of the StorHouse SYSTEM_ID system parameter
type	1 character	Type of component: ■ H for hash index ■ T for table ■ V for value index
component_id	10 decimal digits	Table ID or index ID
segment_id	10 decimal digits	Segment ID containing the file

For example, the following StorHouse file name identifies a table data file in the first segment (segment ID 0) of a user table (table ID150) in the CALLS database.



The following StorHouse file name identifies a value index file in the second segment (segment ID1) of a user table for value index ID 100 in the LONGERTHAN16CHARACTERSDBNAME database.



StorHouse/RM 2.1 to 2.2A segment file names

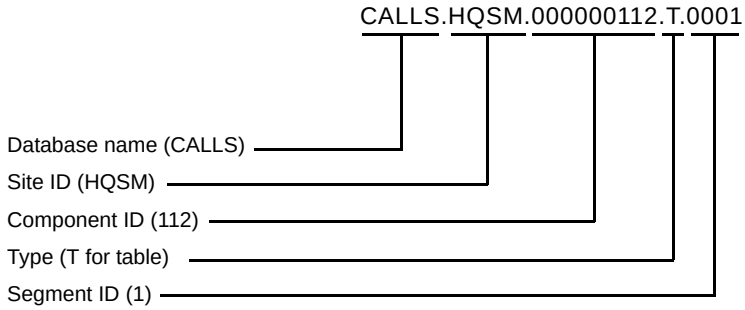
The file naming convention for table data and index files created in StorHouse/RM release 2.1 to 2.2A is as follows:

database_name.site_id.component_id.type.segment_id

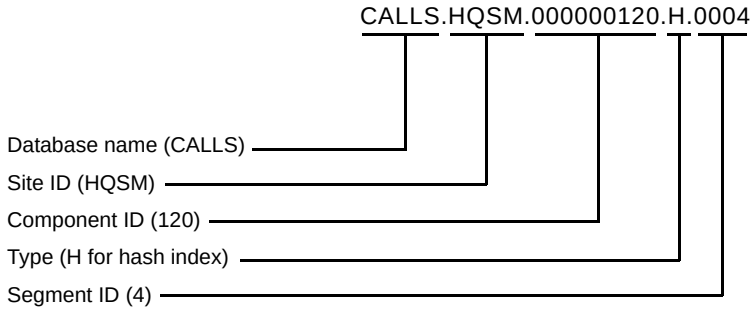
File naming convention for StorHouse/RM 2.1 to 2.2A

File name item	Length	Description
database_name	up to 32 characters	Name of the StorHouse database
site_id	14 characters	Value of the StorHouse SITE_ID system parameter
component_id	9 decimal digits	Table ID or index ID
type	1 character	Type of component: ■ H for hash index ■ T for table ■ V for value index
segment_id	4 decimal digits	Segment ID containing the file

For example, the following StorHouse file name (in the StorHouse/RM release 2.1 to 2.2A format) identifies a table data file in table ID 112. This segment (ID 1) is the second segment in the user table (the first segment has segment ID 0).



The following StorHouse file name identifies a hash index file in index ID 120. This segment (ID 4) is the fifth segment in the user table.



Database system component file names

Each system component in a system tablespace has a UNIX file name. The following table lists the UNIX file naming convention for StorHouse system components. The letter n indicates the table ID or index ID.

UNIX file name format for system components

Component	Name format	Example
System table	Tnnnnnnnnnn.TBL	T0000000012.TBL
System table index	Innnnnnnnnn.IDX	I0000000005.IDX
System table log	Tnnnnnnnnnn.LOG	T0000000012.LOG
System table index log	Innnnnnnnnn.LOG	I0000000005.LOG

Note: When you access system tables through SQL, you use the system table name rather than the UNIX file name. See Appendix A, “StorHouse system tables,” for more information about using system table names in SQL statements.

Database system component identifiers

StorHouse/RM assigns numeric component identifiers (IDs) to system tablespaces, system tables, and system table indexes.

- Tablespace ID 0 is the system tablespace and tablespace ID 1 is the temporary tablespace.
- System table IDs start with 0.
- System table index IDs start with 0.

Summary of StorHouse database concepts

StorHouse databases consist of user components and system components. The following general rules apply to database user components:

- User tables represent data.
- User tables consist of one or more segments.
- A segment is a set of files containing table data, index entries, and LOB data.
- Each file is a STORHOUSE-type file on StorHouse.
- StorHouse/RM supports three types of indexes: value, hash, and range.
- Value and hash indexes are stored in index files.
- Range indexes are stored in system tables.
- You can assign user tables, indexes, and LOB columns to user tablespaces.
- User tablespaces contain subspaces that define how to store table files, index files, and LOB subsegment files on StorHouse.

The following general rules apply to database system components:

- System tables contain data describing a database as well as range index entries.
- Each system table is a separate UNIX file.
- System components are stored in the system tablespace.
- A system tablespace is a database directory in UNIX.
- Each database has a system tablespace.

Chapter

3

StorHouse database administration basics

This chapter describes the basics of StorHouse database administration, such as:

- The roles of people who manage StorHouse databases
- Tools and resources for managing StorHouse databases
- How to perform database administration tasks
- An account for performing database administration tasks

People who manage StorHouse databases

Working with StorHouse databases requires planning and coordination among several people. Because organizations define job responsibilities differently, each person's exact role in StorHouse database administration may vary from site to site. A list of typical roles and responsibilities follows.

- StorHouse chief database administrator (DBA) – Manages all StorHouse databases, creates database user components, and establishes security plans to control access to StorHouse

- StorHouse departmental DBA – Designs and maintains a single StorHouse database, creates database user components, and controls access to database data

- StorHouse system administrator – Works with StorHouse DBAs to develop a storage strategy and to maximize the capabilities of StorHouse

- Application developer – Analyzes data and develops application programs that access data stored in StorHouse
- FileTek customer support representative – Assists with the installation and configuration of StorHouse/RM and provides other support functions as contracted

You in this manual refers to the StorHouse system administrator and to StorHouse DBAs.

Tools to help you manage StorHouse databases

Managing a StorHouse database involves a mix of database administration tasks and system administration tasks. There are two ways to administer StorHouse databases:

- By submitting StorHouse SQL statements and Command Language commands
- With the Control Center GUI-driven StorHouse/Admin module

Note that you must use a StorHouse/Admin version that is compatible with your StorHouse/RM and StorHouse/SM releases.

StorHouse SQL statements

The SQL you use for StorHouse database administration is compatible with the ANSI SQL-99 standard. There are two important differences:

- StorHouse SQL includes extensions to the standard. (An *extension* is an extra option or feature.)
- StorHouse does not support all of the ANSI statements.

Because of these exceptions, the SQL you use to manage a StorHouse database may differ slightly from the SQL you use to manage your local database.

The following table lists the SQL statements you use to manage a StorHouse database.

SQL statements for StorHouse database administration

Use	To
ALTER TABLE SPACE	Change storage specifications in a user tablespace
CREATE INDEX	Create an index for a user table
CREATE SYNONYM	Create a synonym
CREATE TABLE	Create a user table
CREATE TABLE SPACE	Create a user tablespace
CREATE VIEW	Create a view
DELETE	Remove a default tablespace entry for an account
DROP INDEX	Remove an index for a user table
DROP SYNONYM	Remove a synonym
DROP TABLE	Remove a user table
DROP TABLE SPACE	Remove a user tablespace
DROP VIEW	Remove a view
GRANT	Grant privileges to a StorHouse account
INSERT	Assign a default tablespace to an account or database
REVOKE	Remove privileges from a StorHouse account
SELECT	View information in system and user tables
UPDATE	Change the default tablespace for an account or database

Refer to the *StorHouse SQL Reference Manual* for formats and descriptions of these SQL statements. This guide contains examples.

StorHouse Command Language

You or the StorHouse system administrator use StorHouse Command Language commands to complete system administration tasks required for StorHouse/RM. Just as SQL is a set of statements you use to work with a relational database, the *StorHouse Command Language* is a set of commands for managing StorHouse resources. For instance, you use StorHouse commands to create volume sets and file sets for segment files and to create StorHouse accounts for users who will access StorHouse databases. The following table contains some of the StorHouse Command Language commands that you or the StorHouse system administrator may use.

Commands for StorHouse system administration

Use	To
ARCHIVE	Create an archive copy of a segment file or a metadata backup file
CHECKPOINT	Create a checkpoint file of the StorHouse directory after backing up metadata
CREATE ACCOUNT	Create StorHouse accounts
CREATE BACKUP	Create a backup copy of a segment file or a metadata backup file
CREATE FSET	Create file sets for segment files
CREATE VSET	Create volume sets for segment files
REMOVE ACCOUNT	Delete StorHouse accounts
RUN	Invoke the metadata backup utility to back up metadata for one or more StorHouse databases
SET ACCOUNT	Change StorHouse account passwords and change access and command privileges
SET SYSTEM	Set or change StorHouse system parameters
SHOW ACCOUNT	Display StorHouse account IDs and access and command privileges

Commands for StorHouse system administration (continued)

Use	To
SHOW FILE	Display StorHouse file information for table files, index files, LOB subsegment files, and metadata backup files
SHOW FSET	Display information about file sets created for user tablespaces
SHOW SYSTEM	Display values for StorHouse system parameters
SHOW VSET	Display information about volume sets created for user tablespaces

Refer to the StorHouse *Command Language Reference Manual* for complete descriptions of these and other StorHouse commands.

Resources to help you manage StorHouse databases

StorHouse provides the following resources to support database administration: system parameters, user log, and system tables.

System parameters

StorHouse *system parameters* enable you to configure how StorHouse manages various resources or operations. A specific set of system parameters affects StorHouse/RM. For instance, you can specify the maximum number of StorHouse engines that can run concurrently, and you can control the maximum number of concurrent StorHouse engines used to load data.

Some system parameters are informational and others are user-definable. FileTek supplies default values for the user-definable system parameters. You and the StorHouse system administrator should decide whether to use or change the

supplied default values for the system parameters that affect StorHouse/RM. You use the StorHouse Command Language SET SYSTEM and SHOW SYSTEM commands to work with system parameters.

See Appendix C, “System parameters for StorHouse/RM,” for a description of the system parameters you use to tune StorHouse/RM and for instructions on displaying and setting StorHouse system parameters. Refer to the StorHouse *Command Language Reference Manual* for a complete description of all StorHouse system parameters.

User log

To help you monitor system activity, StorHouse records information in the *user log*—a file that provides statistical and administrative information about certain events and various StorHouse activities. You can capture the following StorHouse/RM information in the user log:

- Database connects and disconnects
- Unsuccessful connects (possible security violations)
- SQL statements submitted
- Transaction statistics for each completed transaction, including number of bytes retrieved
- Segment files queried, loaded successfully, and at checkpoint (due to load failure)

You must set various system parameters to enable logging in general. Additionally, whether StorHouse logs SQL, transaction, and file information depends on how you set these system parameters:

- LOG_SQL_STMT
- LOG_SQL_TRANS
- LOG_FILE

Note that the default setting for LOG_SQL_STMT and LOG_SQL_TRANS is TRUE (for logging). Refer to the StorHouse *User Log Format* manual for more information about the user log and the types of events recorded. Refer to the StorHouse *System Administrator's Guide* for more information about how to enable logging.

System tables

StorHouse/RM creates and maintains system tables for each StorHouse database. System tables contain useful information for administration. For example, you can query the SYSTABLES system table to list user table and view names in a database or the SYSSTHSPACES system table to display storage specifications for a user tablespace. You can also maintain the SYSSMUSERS system table to assign default user tablespaces to accounts and a database. See Appendix A, "StorHouse system tables," for more information about StorHouse system tables.

How to perform administration tasks

You perform StorHouse database administration tasks by submitting StorHouse SQL statements. You perform StorHouse system administration tasks by submitting StorHouse Command Language commands.

Submitting StorHouse SQL statements

You can submit SQL to StorHouse with any of the following tools:

- C or C++ programs through StorHouse ESQL
- Local database interactive tools (for example, SPUFI®)
- Local database programs (for example, a DB2 program) through a gateway or a federator
- FileTek data loaders (MVS and FTP)
- StorHouse/Admin Interactive SQL (ISQL)

After you submit an SQL statement, StorHouse/RM generates an SQL status code. Refer to the *StorHouse SQL Reference Manual* for a list of SQL status codes.

Issuing StorHouse commands

You can issue StorHouse Command Language commands with any terminal or workstation that enables you to start a StorHouse session via the Interactive Interface. The StorHouse *Interactive Interface* lets you communicate directly with StorHouse. From your workstation you can sign on to StorHouse, issue StorHouse Command Language commands, and receive StorHouse messages. You can also issue StorHouse Command Language commands with the StorHouse/Admin module of Control Center. Refer to the StorHouse *Command Language Reference Manual* or the *Control Center User Document Set* for more information about issuing StorHouse commands.

Account for performing database and system administration tasks

StorHouse includes one pre-defined account—*SYSADM*—that you can use to perform all StorHouse system and database administration tasks. The initial password for this account is *sysadm*. Be sure to change the password after installation and periodically as needed. See page 4-7 for more information about SYSADM. See page 4-20 for instructions on changing the initial SYSADM account password.

Chapter 4

Accounts and privileges

This chapter describes StorHouse accounts and privileges and explains how to:

- Create an account
- Assign privileges to an account
- Remove privileges from an account
- Change an account password
- List account IDs
- List privileges assigned to accounts
- Remove an account

About StorHouse accounts

A *StorHouse account* with a specified set of StorHouse privileges enables a user to access StorHouse databases. Users need a StorHouse account to:

- Perform StorHouse system and database administration tasks
- Submit StorHouse SQL statements
- Load and unload StorHouse data
- Back up metadata
- Access StorHouse data from a host language application (using ESQL) or a local database application (such as a DB2 UDB or Oracle application)

One StorHouse account can be dedicated to a single user or shared by many users. You create, update, and remove StorHouse accounts with the StorHouse

Command Language CREATE ACCOUNT, SET ACCOUNT, and REMOVE ACCOUNT commands.

Account IDs

When you create a StorHouse account, you assign an account ID. A *StorHouse account ID* consists of 1 to 12 characters that may include A–Z (uppercase), 0–9, $ (dollar sign), and _ (underscore). StorHouse always converts account IDs to uppercase. FileTek recommends that account IDs follow StorHouse SQL identifier naming conventions, but you can delimit account IDs that do not follow those conventions. Refer to the *StorHouse SQL Reference Manual* for more information about specifying delimited account IDs in SQL statements.

Account passwords

You assign an account password when you create a StorHouse account. An *account password* can contain up to 32 characters consisting of the following ASCII characters: A–Z (uppercase), 0–9, $ (dollar sign) and _ (underscore). StorHouse always converts account passwords to uppercase.

When you assign a password, that account must always specify it when:

- Logging into the StorHouse FTP server to load data with the FileTek FTP Data Loader or to unload data with the FileTek FTP Data Unloader
- Connecting to a StorHouse database (in the USING clause of a CONNECT statement)
- Supplying the ACCT value on the SMDEF control statement to load data with the FileTek MVS Data Loader utility

If you don't assign a password, then the account cannot access StorHouse databases.

About StorHouse privileges

StorHouse provides a complete set of privileges—access, command, database, and database component—that defines different functions an account can perform. You assign access and command privileges when creating a StorHouse account. These privileges define the functions an account can perform in all StorHouse databases. You can then grant database and database component privileges to that account. These privileges define the functions an account can perform in a specific StorHouse database.

While there are many StorHouse access and command privileges, a subset of these privileges affects StorHouse/RM. Refer to the StorHouse *Command Language Reference Manual* for a complete list and definition of all access and command privileges. The following table summarizes the access and command privileges and the database and database component privileges for StorHouse/RM.

StorHouse privileges

Type of privilege	Privileges
Access	SQLADMIN
Command	SQLEXECUTE and SQLCOMMAND
Database	DBA, RESOURCE, and SCAN
Database component	ALL, DELETE, INDEX, INSERT, SELECT, and UPDATE

Access privilege

The *SQLADMIN access privilege* enables an account to perform database administration tasks in *all* StorHouse databases. Specifically, SQLADMIN gives an account the DBA database privilege in all StorHouse databases. To actually perform database administration tasks, however, the account must also have the SQLEXECUTE command privilege. A chief DBA, for example, might have this SQLADMIN privilege.

Command privileges

Command privileges enable accounts to submit certain StorHouse commands. While there are many StorHouse command privileges, two command privileges are specific to StorHouse databases: SQLEXECUTE and SQLCOMMAND.

SQLEXECUTE privilege

The *SQLEXECUTE command privilege* enables an account to submit StorHouse SQL statements in all StorHouse databases. An account must have this privilege at a minimum to access StorHouse databases. Although accounts with this privilege can submit SQL statements in *all* StorHouse databases, database and database component privileges restrict access to components in specific databases.

SQLCOMMAND privilege

The *SQLCOMMAND command privilege* enables an account to run a FileTek data loader and unloader in all StorHouse databases. Specifically, this privilege permits an account to invoke a data loader and unloader, which then submits a StorHouse Command Language EXECUTE STH_LOAD command. The SQLCOMMAND privilege is just one of several privileges required to load and unload data. See “Loader accounts” on page 4-8 for a list of the privileges required to load data. See “Unloader accounts” on page 4-9 for a list of the privileges required to unload data.

Database privileges

Database privileges control the functions an account can perform in a specific database. There are three StorHouse database privileges:

- DBA
- RESOURCE
- SCAN

If an account has none of these database privileges, then that account is a *general database user account* and has database component privileges on the database only.

DBA privilege

DBA privilege is the highest privilege an account can have on a specific database. Typically, this privilege is reserved for a departmental DBA. (An account with the SQLADMIN access privilege has DBA privilege on all databases.) An account with DBA privilege can perform the following functions for a specific database:

- Create database user components (user tables, indexes, views, and synonyms)
- Create database user components for other accounts
- Access and maintain system tables
- Grant database and database component privileges to accounts
- Revoke database and database component privileges from accounts
- Select data from any table, view, or synonym

DBA privilege includes all the privileges provided by the RESOURCE privilege. It does not include SCAN privilege; however, an account with DBA can grant SCAN privilege to other accounts.

RESOURCE privilege

An account with RESOURCE privilege has more database privileges than a general database user account but fewer privileges than an account with DBA privilege. Such an account can do the following for a specific database:

- Create database user components
- Select information from these database user components
- Grant other accounts SELECT and INDEX privileges on these components

SCAN privilege

An account with SCAN privilege can read all rows in any user table (in other words, perform a full table scan on a user table) on which it has SELECT privilege. The purpose of SCAN privilege is to control which accounts can do full table scans. Because full table scans can be very time consuming, you may want to limit their use.

Accounts that use the StorHouse *extractor* feature to perform full table scans require the SCAN privilege. Accounts that use the FileTek FTP Data Unloader also require SCAN when unloading an entire table or when the unload operation must read all rows in a table without the use of an index.

Database component privileges

Database component privileges determine an account's access to specific database components, such as tables and views or columns within tables and views. Column level privileges apply to system tables only because StorHouse does not allow updates of user tables. StorHouse database component privileges are as follows.

Database component privileges

Privilege	Authorizes the account to
ALL	Have all privileges on a specific table or view
DELETE	Delete rows from a system table or system table view
INDEX	Create an index on a user table
INSERT	Load data into a user table or insert rows into a system table or view
SELECT	Access data in a table or view
UPDATE	Update one or more columns in a system table or system table view

The following rules apply to database component privileges:

- An account with DBA privilege on a database has all available database component privileges on all components.
- An account with RESOURCE privilege on a database has all available database component privileges on any components created by that account.
- General database user accounts have those database component privileges granted to them by other accounts and those available to PUBLIC. If they've been granted no database component privileges by other accounts, they have PUBLIC privileges only.
- No matter what an account's database component privileges, the account has ALL component privileges on any table or view that it owns.

Types of accounts and their privileges

This section describes two special accounts—SYSADM and PUBLIC—as well as the combination of privileges required to create other types of accounts.

SYSADM account

SYSADM, a pre-defined StorHouse account, has all StorHouse privileges. This account provides immediate full access to StorHouse, including every StorHouse database. Using SYSADM, you can:

- Perform StorHouse system administration tasks
- Perform all StorHouse database administration tasks on all databases
- Run the metadata backup utility
- Submit SQL statements in all StorHouse databases
- Do full table scans on all user tables in all StorHouse databases
- Grant SCAN database privilege to other accounts

- Run a FileTek data loader to load data into user tables
- Unload data with the FileTek FTP Data Unloader

PUBLIC account

PUBLIC is a term for the collective group of all StorHouse accounts. You can grant only database component privileges to PUBLIC. You can't grant access, command, and database privileges to PUBLIC. By granting a database component privilege to PUBLIC, you grant it to every account with SQLEXECUTE privilege.

The privileges available to PUBLIC vary from database to database. See page 4-12 for more information about granting and revoking privileges to and from PUBLIC. You can also use PUBLIC to designate a default user tablespace for a database. See page 5-28 for more information about designating a default user tablespace for a database.

General database user accounts

StorHouse accounts with SQLEXECUTE privilege and without database privileges (DBA, RESOURCE, or SCAN) are general database user accounts. These accounts have all privileges granted to PUBLIC and any database component privilege granted to the account.

Loader accounts

To load data into StorHouse user tables, load deferred indexes, and merge segments, a StorHouse account requires a combination of StorHouse command and database component privileges. The minimum command privileges needed to run a FileTek data loader are as follows:

- ATF
- DELETE
- GET
- PUT
- RECORD
- SETGROUP
- SQLCOMMAND
- SQLEXECUTE
- VTF

The account must also have the INSERT database component privilege on the user tables it is loading. See “Loader accounts” on page 4-16 for an example loader account.

Unloader accounts

To unload data using the FileTek FTP Data Unloader, a StorHouse account requires these privileges:

- SELECT database component privilege on the user table(s) to be unloaded
- SCAN database privilege (if the query results in a full table scan)
- SQLEXECUTE command privilege to submit StorHouse SQL
- SQLCOMMAND command privilege to invoke the unloader

Metadata backup accounts

To run the metadata backup utility, a StorHouse account requires one of the following StorHouse command privileges:

- OPERATOR
- SERVICE
- SYSTEM

The SYSADM account can run the metadata backup utility because that account has all command privileges. You must assign the OPERATOR, SERVICE, or SYSTEM privilege to authorize any other account to run the metadata backup utility. Refer to the StorHouse *Command Language Reference Manual* for more information about these command privileges.

Local database application accounts

To access StorHouse from a local database application (for example, a DB2 or Oracle application), you must create a StorHouse account for the application user. For instance, DB2 UDB users require StorHouse accounts to access StorHouse databases. Those accounts must be mapped to DB2 authorization

IDs. Refer to the *StorHouse/UDB Link Installation and Configuration Manual* for more information about managing account security and privileges in a federated system.

At a minimum, a local database application account must have SQLEXECUTE privilege. Assign other StorHouse privileges as appropriate. For instance, if the account will create user tables and indexes, then assign the RESOURCE privilege.

Host language application accounts

To access StorHouse from a host language application, you create a StorHouse account and specify the account ID in the USER clause and the account password (expressed as a host variable) in the USING clause of your ESQL program CONNECT statement. At a minimum, a host language application account must have SQLEXECUTE privilege. Assign other StorHouse privileges as appropriate. For instance, if the account may perform full table scans, assign the SCAN privilege and the SELECT privilege on all applicable user tables.

Ways to assign privileges

You assign access and command privileges when you create or update accounts with the StorHouse Command Language CREATE ACCOUNT or SET ACCOUNT commands. You assign database and database component privileges with the StorHouse SQL GRANT and REVOKE statements. The GRANT statement grants a database or database component privilege to an account, while the REVOKE statement removes such a privilege from an account.

The following rules apply to granting and revoking database and database component privileges:

- An account with DBA privilege on a given database can grant privileges on that database to any StorHouse account.
- An account with RESOURCE privilege on a database can grant privileges on only those database components that it owns to any StorHouse account.

- If you are authorized to grant or revoke privileges, you can do so for any StorHouse account except your own (you can't grant to or revoke from yourself).
- New privileges take effect with the next request following execution of the GRANT or REVOKE statement.
- Only accounts with the SQLEXECUTE privilege can use their privileges. For example, if you grant SCAN privilege to an account but that account does not have SQLEXECUTE, then the account cannot scan user tables.

As mentioned, you grant database and database component privileges using the GRANT statement. The following example grants SELECT privilege on the SALARY table to USER1:

```
GRANT SELECT ON SALARY
TO USER1
```

The WITH GRANT OPTION

The WITH GRANT OPTION argument on the GRANT statement enables accounts to grant their privileges, or a subset of their privileges, to other accounts. If you omit the argument, the account to which you're granting privileges (grantee) cannot grant those privileges to other accounts.

For example, to grant USER1 the SELECT privilege on the SALARY table with the WITH GRANT OPTION argument, issue the following statement:

```
GRANT SELECT ON SALARY
TO USER1
WITH GRANT OPTION
```

Now USER1 can grant the SELECT privilege on the SALARY table to other accounts.

When you grant to PUBLIC

You can grant and revoke database component privileges to and from PUBLIC. By doing this, you can grant or revoke privileges to all accounts with just one GRANT or REVOKE statement.

For instance, suppose you just created a new table called RECORDS and you'd like to grant SELECT privilege on that table to all StorHouse accounts. Instead of submitting individual GRANT statements for each account, you can grant SELECT privilege on the RECORDS table to PUBLIC in one statement as follows:

```
GRANT SELECT ON RECORDS
TO PUBLIC
```

Now all StorHouse accounts with SQLEXECUTE privilege can select information from RECORDS.

You can also revoke privileges from PUBLIC. For instance, should you need to restrict access to RECORDS from all accounts but three, you'd revoke SELECT privilege from PUBLIC and then grant SELECT privilege to the three accounts that require the privilege. You'd issue the following statement to revoke SELECT from PUBLIC:

```
REVOKE SELECT ON RECORDS
FROM PUBLIC
```

Remember the following:

- PUBLIC privileges vary from database to database. For each database there is a separate set of privileges available to PUBLIC.
- Any StorHouse account with SQLEXECUTE has PUBLIC access to all StorHouse databases.
- An account's privileges are the combination of those granted to PUBLIC and those granted to the account.

- If PUBLIC has SELECT privilege on a table, then all accounts have SELECT privilege on the table. Revoking SELECT from individual accounts when PUBLIC has SELECT has no effect.

System tables with privilege information

StorHouse/RM records database and database component privileges in three system tables. StorHouse/RM updates these system tables when you grant and revoke privileges.

- SYSDBAUTH – Contains accounts and their database privileges (DBA, RESOURCE, and SCAN).
- SYSCOLAUTH – Contains accounts with the UPDATE database component privilege on system table columns.
- SYSTABAUTH – Contains accounts and their database component privileges (DELETE, INDEX, INSERT, SELECT, and UPDATE) on tables and views.

Summary of StorHouse privileges

StorHouse privileges define the functions an account can perform and the database components an account can access. In summary:

- Four types of StorHouse privileges are access, command, database, and database component.
- Access (SQLADMIN) and command (SQLEXECUTE and SQLCOMMAND) privileges enable an account to perform specific functions in all StorHouse databases.
- Database (DBA, RESOURCE, and SCAN) and database component (ALL, DELETE, INDEX, INSERT, SELECT, and UPDATE) privileges enable an

account to perform specific functions and access certain components in specific StorHouse databases.

- You assign an access or command privilege when you create an account. You create accounts using the StorHouse Command Language CREATE ACCOUNT command.
- You assign a database or database component privilege by using the StorHouse SQL GRANT statement.
- You remove or change an access or command privilege by using the StorHouse Command Language SET ACCOUNT command.
- You remove a database or database component privilege by using the StorHouse SQL REVOKE statement.

Creating a StorHouse account

You can create a StorHouse account by submitting the StorHouse Command Language CREATE ACCOUNT command. You must assign one or more of the access and command privileges—SQLADMIN, SQLEXECUTE, and SQLCOMMAND—to the account. You must also assign an account password.

Keep in mind the following when creating a StorHouse account:

- Assign an account ID that conforms to SQL identifier naming conventions. See “Account IDs” on page 4-2 for more information about naming conventions.
- If you don’t assign an account password, then the account cannot access StorHouse databases.
- If you disable an account with the /DISABLED parameter modifier, the account cannot access StorHouse database.

- To give the account complete DBA access to all StorHouse databases, assign the SQLADMIN privilege. An account with SQLADMIN, however, also needs SQLEXECUTE to perform database tasks.

- To enable any account to access StorHouse databases, assign the SQLEXECUTE privilege. The account's access to individual databases is then determined by database and database component privileges granted to the account and database component privileges granted to PUBLIC.

- To enable an account to run the FileTek MVS Data Loader utility, the FileTek FTP Data Loader, or the FileTek FTP Data Unloader, assign the SQLCOMMAND privilege. You must also assign other command privileges and grant database component privileges as listed at "Loader accounts" on page 4-8 and "Unloader accounts" on page 4-9.

- To enable an account to run the metadata backup utility, assign the OPERATOR, SERVICE, or SYSTEM privilege.

- To enable an account to perform full table scans or to use the extractor feature, assign the SCAN privilege.

▼ To create a StorHouse account

Required privileges: ACCOUNT, ANYACCOUNT, and SETGROUP

1. Sign on to StorHouse.

2. At the command prompt (?), submit a StorHouse Command Language CREATE ACCOUNT command and press J . Some examples follow.

   – To create the account USER1 with a password of SPRING and with SQLEXECUTE command privilege, type:

```
CREATE ACCOUNT USER1 SPRING /PRIVILEGE=SQLEXECUTE
```

– To create the account USER2 with a password of SUMMER and with SQLADMIN access privilege and SQLEXECUTE command privilege, type:

```
CREATE ACCOUNT USER2 SUMMER /PRIVILEGE=(SQLADMIN,
SQLEXECUTE)
```

– To create the account LOADER with a password of FALL that can load data and run the metadata backup utility, type:

```
CREATE ACCOUNT LOADER FALL /PRIVILEGE=(OPERATOR,
SQLEXECUTE, SQLCOMMAND, RECORD, PUT, GET, SETGROUP,
DELETE, ATF, VTF)
```

Granting database privileges

You can grant the DBA, RESOURCE, or SCAN database privilege by submitting a GRANT statement. StorHouse accounts require SQLEXECUTE privilege to use their database privileges.

▼ To grant database privileges to a StorHouse account

Required privileges: SQLEXECUTE and DBA

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a GRANT statement to grant a database privilege. Some examples follow.

   – To grant DBA and SCAN privileges to USER1, type:

```
GRANT DBA, SCAN
TO USER1
```

– To grant RESOURCE privilege to USER2, type:

```
GRANT RESOURCE
TO USER2
```

Granting database component privileges

You can grant database component privileges to accounts by submitting a GRANT statement. StorHouse accounts require SQLEXECUTE privilege to use their database component privileges. Database component privileges are:

- ALL
- DELETE (only for system tables and system table views)
- INDEX
- INSERT (only for system tables, system table views, and loading user tables)
- SELECT
- UPDATE (only for system tables and system table views)

▼ To grant a database component privilege to a StorHouse account

Required privileges: SQLEXECUTE and DBA or component owner

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a GRANT statement to grant a database component privilege. Some examples follow.

   – To grant INDEX privilege on the CUSTOMERS table to USER1, type:

```
GRANT INDEX
ON CUSTOMERS
TO USER1
```

– To grant SELECT privilege on the CUSTOMERS table to USER1 and grant USER1 the WITH GRANT OPTION to grant SELECT privilege to other accounts, type:

```
GRANT SELECT
ON CUSTOMERS
TO USER1
WITH GRANT OPTION
```

Revoking database privileges

You can revoke the DBA, RESOURCE, or SCAN database privilege from an account by submitting a REVOKE statement. See "Listing database privileges" on page 4-24 to verify an account's database privileges. If you're removing all database privileges for an account, you can specify all the privileges in one REVOKE statement.

▼ To revoke database privileges from a StorHouse account

Required privileges: SQLEXECUTE and DBA

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a REVOKE statement to revoke a database privilege from an account. Some examples follow.

   – To revoke the DBA privilege from USER1, type:

```
REVOKE DBA
FROM USER1
```

– To revoke all three database privileges from USER2, type:

```
REVOKE DBA, RESOURCE, SCAN
FROM USER2
```

Revoking database component privileges

You can revoke database component privileges—ALL, DELETE, INDEX, INSERT, SELECT, and UPDATE—from an account by submitting a REVOKE statement. See "Listing database component privileges" on page 4-25 to verify an account's database component privileges. If you're removing all database component privileges for an account, you can revoke all privileges in one REVOKE ALL statement.

Note: You cannot revoke a database component privilege from that component's owner.

▼ To revoke database component privileges from a StorHouse account

Required privileges: SQLEXECUTE and DBA or component owner

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a REVOKE statement to revoke a database component privilege from an account. Some examples follow.

   – To revoke the INDEX privilege on the CUSTOMERS table from USER1, type:

```
REVOKE INDEX
ON CUSTOMERS
FROM USER1
```

– To revoke all privileges on the CUSTOMERS table from USER1, type:

```
REVOKE ALL
ON CUSTOMERS
FROM USER1
```

Changing a StorHouse account password

You can change an account password by using the StorHouse Command Language SET ACCOUNT command. The new password takes effect the next time the account accesses StorHouse.

▼ To change an account password

Required privileges: ACCOUNT, ANYACCOUNT, PASSWORD, and SETGROUP

1. Sign on to StorHouse.

2. At the command prompt (?), submit the StorHouse Command Language SET ACCOUNT command with the /NEWPASSWORD parameter modifier and press J . An example follows.

   – To change USER1's password, type the following command:

   ```
   SET ACCOUNT USER1 /NEWPASSWORD
   ```

   The system responds:

   ```
   Enter new password or RETURN for none?
   Reenter new password or RETURN for none?
   ```

   Enter the new password in response to each prompt. The passwords entered do not display on the screen.

Adding an access or command privilege

You can add an access or command privilege to a StorHouse account by using the StorHouse Command Language SET ACCOUNT command. The new privilege takes effect the next time the account accesses StorHouse.

▼ To add an access or command privilege to a StorHouse account

Required privileges: ACCOUNT, ANYACCOUNT, and SETGROUP

1. Sign on to StorHouse.
2. At the command prompt (?), submit the appropriate StorHouse Command Language SET ACCOUNT command and press J . Some examples follow.
   - To add SQLCOMMAND to USER1's StorHouse privileges, type:

     SET ACCOUNT USER1 /PRIVILEGE=SQLCOMMAND
   - To add SQLADMIN and SQLCOMMAND to USER3's StorHouse privileges, type:

     SET ACCOUNT USER3 /PRIVILEGE=(SQLADMIN,SQLCOMMAND)
   - To add OPERATOR to USER2's StorHouse privileges, type:

     SET ACCOUNT USER2 /PRIVILEGE=OPERATOR

Removing an access or command privilege

You can remove an access or command privilege from a StorHouse account by using the StorHouse Command Language SET ACCOUNT command and preceding the privilege to be removed with NO. Removed StorHouse access and command privileges take effect the next time the account accesses StorHouse.

▼ To remove an access or command privilege from a StorHouse account

Required privileges: ACCOUNT, ANYACCOUNT, and SETGROUP

1. Sign on to StorHouse.

2. At the command prompt (?), submit the appropriate StorHouse Command Language SET ACCOUNT command and press J . Some examples follow.

   – To remove SQLCOMMAND from USER1's StorHouse privileges, type:

     SET ACCOUNT USER1 /PRIVILEGE=NOSQLCOMMAND

   – To remove SQLADMIN and SQLCOMMAND from USER2's StorHouse privileges, type:

     SET ACCOUNT USER2 /PRIVILEGE=(NOSQLADMIN, NOSQLCOMMAND)

Listing StorHouse account IDs

You can list all StorHouse accounts that can access StorHouse databases by using the StorHouse Command Language SHOW ACCOUNT command.

▼ To list all StorHouse accounts that can access StorHouse databases

Required privilege: SHOW

1. Sign on to StorHouse.

2. At the command prompt (?), submit the following StorHouse Command Language SHOW ACCOUNT command, and then press J .

```
SHOW ACCOUNT * /PRIV=(SQLADMIN,SQLEXECUTE,
SQLCOMMAND, ALLPRIV)
```

Listing access and command privileges

You can list access and command privileges for one or more StorHouse accounts by using the StorHouse Command Language SHOW ACCOUNT command.

▼ To list access and command privileges for a StorHouse account

Required privilege: SHOW

1. Sign on to StorHouse.
2. At the command prompt (?), submit the appropriate StorHouse Command Language SHOW ACCOUNT command and press J . Some examples follow.
   - To list access and command privileges for a specific account, such as USER1, type:

     ```
     SHOW ACCOUNT USER1 /PRIVILEGE
     ```

   - To list access and command privileges for all StorHouse accounts, type:

     ```
     SHOW ACCOUNT * /PRIVILEGE
     ```

Listing the privileges granted to PUBLIC

StorHouse accounts with SQLEXECUTE privilege automatically get all the database component privileges granted to PUBLIC for each StorHouse database.

You can query the SYSTABAUTH system table to determine the privileges granted to PUBLIC.

▼ To list the privileges granted to PUBLIC

Required privileges: SQLEXECUTE and either DBA or SELECT on SYSTABAUTH

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a SELECT statement on the SYSTABAUTH system table. Remember to precede the system table name with SYSADM (the system table owner name). Use uppercase when specifying the grantee PUBLIC. For example:

```
SELECT *
FROM SYSADM.SYSTABAUTH
WHERE GRANTEE = 'PUBLIC'
```

Listing database privileges

You can list database privileges—DBA, RESOURCE, or SCAN—for one or more StorHouse accounts by submitting a SELECT statement on the SYSDBAUTH system table.

▼ To list StorHouse database privileges for a StorHouse account

Required privileges: SQLEXECUTE and either DBA or SELECT on SYSDBAUTH

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a SELECT statement to list database privileges. Some examples follow.

– To list all StorHouse database privileges for all accounts, type:

```
SELECT *
FROM SYSADM.SYSDBAUTH
```

A y in a column indicates the account has the privilege. A g indicates the account has the privilege with the WITH GRANT OPTION.

– To list all StorHouse database privileges for grantee (account) USER1, type:

```
SELECT DBA_ACC, RES_ACC, SCN_ACC
FROM SYSADM.SYSDBAUTH
WHERE GRANTEE='USER1'
```

A y in a column indicates USER1 has the privilege. A g indicates USER1 has the privilege with the WITH GRANT OPTION.

– To list all StorHouse accounts with DBA privilege, type:

```
SELECT GRANTEE
FROM SYSADM.SYSDBAUTH
WHERE DBA_ACC='y'
```

Listing database component privileges

You can list database component privileges—ALL, DELETE, INDEX, INSERT, SELECT, and UPDATE—for one or more StorHouse accounts by submitting a SELECT statement on the SYSCOLAUTH and SYSTABAUTH system tables, respectively.

The following rules apply to database component privileges:

- The INDEX and SELECT table privileges apply to both user tables and system tables.
- The DELETE and UPDATE privileges apply to system tables only.
- The INSERT privilege applies to system tables only; however, the account used by a FileTek data loader to sign on to StorHouse requires INSERT privilege to load data into user tables.

▼ To list StorHouse database component privileges for a StorHouse account

Required privileges: SQLEXECUTE and either DBA or SELECT on both SYSTABAUTH and SYSCOLAUTH

1. Follow your site's procedure for connecting to your StorHouse database.
2. Submit a SELECT statement to list account database component privileges. Some examples follow.
   - To list all columns for which the grantee (account) USER1 has UPDATE privilege and to list the system table in which each column resides, type:

     ```
     SELECT TBL, COL
     FROM SYSADM.SYSCOLAUTH
     WHERE GRANTEE='USER1'
     AND UPD='y'
     ```

– To list all StorHouse table and view privileges for grantee (account) USER1 and to list the table or view on which each privilege was granted, type:

```
SELECT TBL, INS, DEL, UPD, SEL, NDX, ALT
FROM SYSADM.SYSTABAUTH
WHERE GRANTEE='USER1'
```

A y in a column indicates the account has the appropriate privilege. A g indicates the account has the privilege with the WITH GRANT OPTION.

Removing a StorHouse account

You can remove a StorHouse account by performing the following tasks:

- Task 1 – For each database, revoke all database component privileges granted to the account.
- Task 2 – For each database, revoke all database privileges granted to the account.
- Task 3 – Delete the StorHouse account.

Because you're removing the account, you can revoke all privileges on a database component with one REVOKE ALL statement and all database privileges with one REVOKE statement. You must issue these statements for all databases that the account can access.

▼ To remove a StorHouse account

Required privileges: ACCOUNT, ANYACCOUNT, SQLEXECUTE, and DBA

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a REVOKE ALL statement for each database component. For example, if USER1 has database component privileges on the CUSTOMERS table and the SALARIES table, then to revoke all database component privileges on those tables, submit this statement:

   ```
   REVOKE ALL
   ON CUSTOMERS
   FROM USER1
   ```

   and then submit this statement:

   ```
   REVOKE ALL
   ON SALARIES
   FROM USER1
   ```

3. Submit a REVOKE statement to revoke all database privileges from the account. For example, to revoke all database privileges from USER1, type:

   ```
   REVOKE DBA, RESOURCE, SCAN
   FROM USER1
   ```

4. Sign on to StorHouse.

5. At the command prompt (?), submit the StorHouse Command Language REMOVE ACCOUNT command and press J . For example, to remove the account USER1, type:

   ```
   REMOVE ACCOUNT USER1
   ```

   If USER1 is signed on when the command is executed, the current session is not affected. However, after USER1's session is terminated, the account ID USER1 cannot be used again to access StorHouse.

Chapter 5

User tablespaces

This chapter describes user tablespaces and explains how to:

- Create a user tablespace
- Assign and change a default user tablespace
- List accounts and their default user tablespaces
- Display default storage specifications
- Alter a user tablespace
- List user tablespace names and IDs
- Display storage specifications for a user tablespace
- Drop a user tablespace

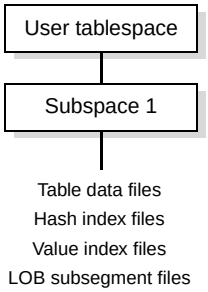
About user tablespaces

Before creating and loading user tables in a database, you create one or more user tablespaces and can designate a default user tablespace for the database. A *user tablespace* is a logical database component that defines where to store segment files on StorHouse and how to manage them. When you create a user table, you assign it to a user tablespace. You can assign LOB columns to the same user tablespace as the table or to different user tablespaces. And when you create indexes for that user table, you can assign them to the same user tablespace as the table or to different ones. The table files, index files, and LOB subsegment files are stored according to the specifications of their assigned user tablespace.

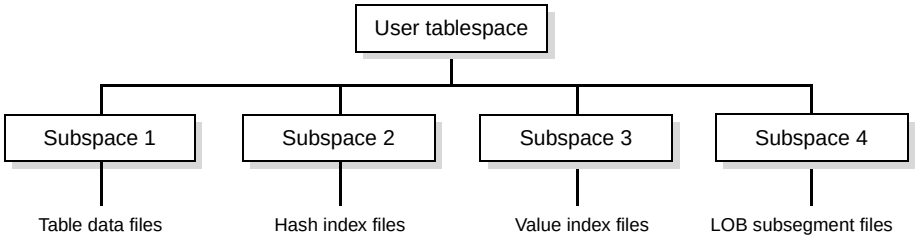
You create user tablespaces with the CREATE TABLE SPACE statement. See page 5-26 for a summary of the CREATE TABLE SPACE arguments.

Subspaces

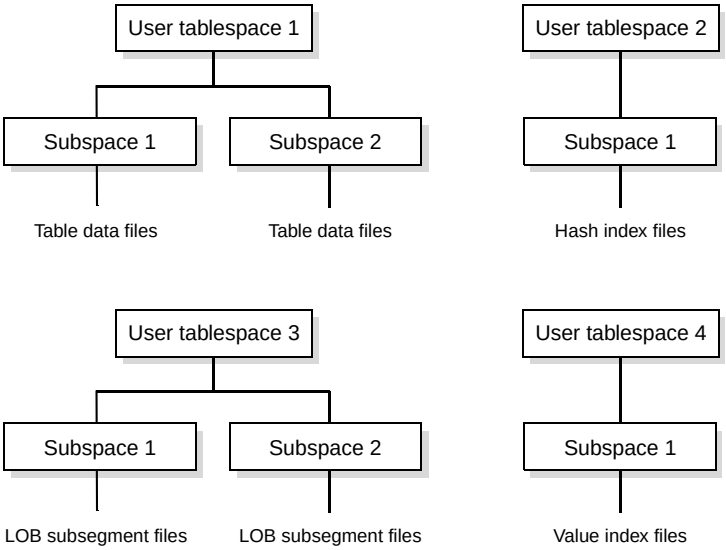
A user tablespace consists of one or more subspaces. A *subspace* contains a set of storage specifications for a specific type of user table component—table data, indexes, LOB data—or for all components. For instance, to manage the storage of all components in the same way, you can create a user tablespace with one subspace—or one set of storage specifications—for all components.



Or to manage the storage of different components in different ways, you can define multiple subspaces in a user tablespace. For instance, you can define one subspace for table data files, a second subspace for hash index files, a third subspace for value index files, and a fourth subspace for LOB subsegment files, defining different storage specifications for each subspace.



You can also assign all components to different user tablespaces. Each user tablespace can consist of one or more subspaces.



Whether you use multiple user tablespaces and/or multiple subspaces depends on your storage strategy. The following section, “Storage specifications,” defines the storage options that you set in a subspace. See “User tablespace examples” on page 5-12 for examples.

Note: When space permits or by user request, StorHouse/RM stores in-line LOBs with the table data, ignoring any LOB subspace or user tablespace. Subspaces for LOB subsegment files apply to out-of-line LOBs only. See “LOB storage and subsegment files” on page 2-10 for more information about in-line and out-of-line LOBs.

Storage specifications

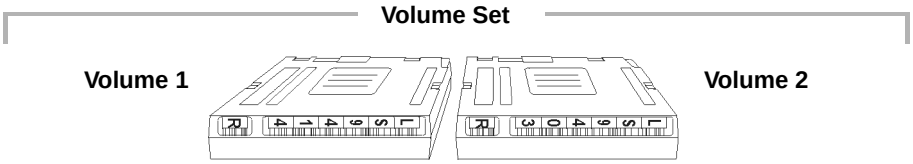
You define *storage specifications* for each subspace in a user tablespace. You can change those specifications when needed. Storage specifications are:

- Volume set
- File set
- Object type
- Access Time Factor
- Vulnerability Time Factor
- Error Detection Code
- File access group
- Maximum extent size
- Extent retention

The volume set and file set are required for each subspace. You can use default values, if applicable, for the remaining storage specifications.

Volume set

A *volume set* (VSET) is one or more physical volumes that are treated as a logical unit of storage. A *volume* is a unit of media, such as an optical disk cartridge or a tape cartridge, on which data can be recorded and read. For example, the following volume set consists of two tape cartridges.

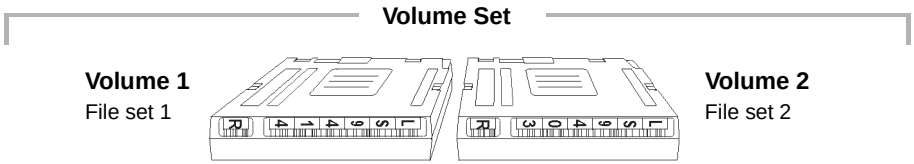


When you create a volume set, you specify a library device and the media type. You also specify volume set properties, such as the initial size to allocate and whether there's a limit, if any, to how large the volume set can grow. Refer to the *StorHouse Concepts and Facilities Manual* and the StorHouse *System*

Administrator's Guide for more information about volume sets and their properties.

File set

A *file set* (FSET) is a a collection of files within a volume set. Typically, a volume set contains a single file set, but it can contain multiple file sets. For example, the following volume set consists of two file sets. You could, for instance, store table data files and LOB subsegment files in file set 1 and index files in file set 2.



Whether you use multiple volume sets and file sets for segment files depends on your data and your access and performance requirements. Refer to the StorHouse *System Administrator's Guide* for guidelines on using one or more volume sets and file sets.

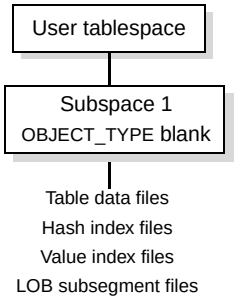
Object type

The *object type* defines the type of component to be stored in a subspace. The object type values are:

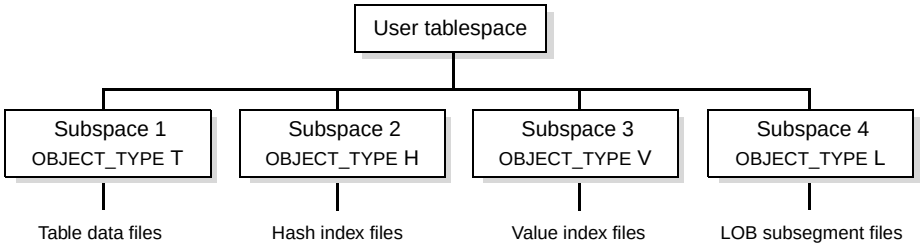
Object type values in a subspace

OBJECT_TYPE values	Description
Blank	All component types
T	Table data only
H	Hash indexes only
V	Value indexes only
L	LOB data only

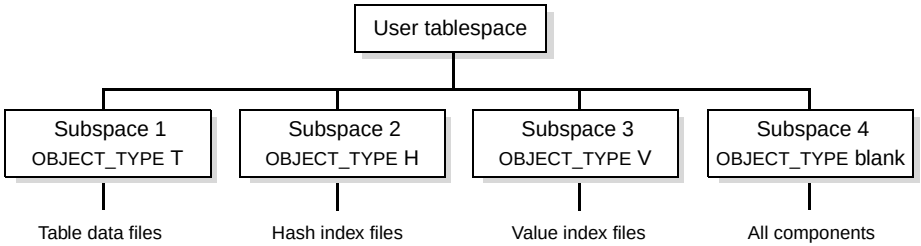
For instance, you can allow all components in a subspace.



Or you can allow only certain components in a subspace.



Or you can define a mix of subspaces that allows all components and specific components.



When a user tablespace contains multiple subspaces for a component type, a FileTek data loader automatically selects the lowest-numbered subspace for each component type. For instance, in the preceding example:

- Subspace 1 and subspace 4 allow table data files and in-line LOBs. The data loader automatically selects subspace 1 for table data and in-line LOBs.
- Subspace 2 and subspace 4 allow hash index files. The data loader automatically selects subspace 2 for hash indexes.
- Subspace 3 and subspace 4 allow value index files, The data loader automatically selects subspace 3 for value indexes.
- Subspace 4 is the only subspace that allows out-of-line LOBs. The data loader automatically selects subspace 4 for out-of-line LOBs.

During a data load, index load, or segment merge operation, however, you can select subspaces or request to rotate among subspaces if you do not want to use the lowest-numbered subspace for a component type. In this example, for instance, you can select subspace 4 for all components, or you can select subspace 4 for just table data and LOB data and continue to use the applicable lowest-numbered subspace for each index type. Refer to the *FileTek FTP Data Loader Manual* or the *FileTek MVS Data Loader Utility Manual* for more information about subspace selection and rotation during data load, index load, and merge operations.

Access Time Factor

The *Access Time Factor* (ATF) indicates the importance of access time for segment files. This parameter works with the StorHouse migrate function to keep data that is most likely to be accessed in the StorHouse performance buffer while

maintaining a supply of free space. You can assign one ATF value for each subspace. The ATF values are:

ATF values in a subspace

ATF values	Description
0	Use the value of the StorHouse ATF system parameter, which sets the ATF value to the system default.
1	Short access time is very important.
2	Short access time is moderately important.
3	Short access time is less important.

Refer to the *StorHouse Concepts and Facilities Manual* for more information about ATF and migration.

Vulnerability Time Factor

The *Vulnerability Time Factor* (VTF) determines when StorHouse writes files to their resident file sets as well as whether to create performance copies of files in the performance buffer. You can assign one VTF value for each subspace. The VTF values are:

VTF values in a subspace

VTF values	Description
DIRECT	Bypass the performance buffer and directly write the files to their file sets. (DF and map extents, however, are always written to the performance buffer as well as to their resident file set.)
NOW	Write extents to the performance buffer first and copy them to their file sets second. StorHouse/RM copies each extent to the resident file set after creating the extent on the performance buffer. So for a table file, StorHouse/RM creates the data extent on the performance buffer and then copies the data extent to its resident file set. Then StorHouse/RM creates the DF extent on the performance buffer and then copies it to the file set. Both of these operations (create and copy) take place during the load.

VTF values in a subspace (continued)

VTF values	Description
NEXT	Write extents to the performance buffer during a load and copy them to their file sets during the next StorHouse write-back operation.
DEFAULT	Use the value of the StorHouse VTF system parameter, which sets the VTF value to the system default.

Refer to the *StorHouse Concepts and Facilities Manual* for more information about VTF, the performance buffer, and write-back operations.

Error Detection Code

The *Error Detection Code* (EDC) parameter determines whether to use the StorHouse error detection feature for segment files. StorHouse generates EDCs during data loads and uses EDCs to detect errors during data movement in StorHouse. FileTek recommends that you always use EDC. You can assign one EDC value for each subspace. The EDC values are:

EDC values in a subspace

EDC values	Description
Y	Use EDC.
N	Do not use EDC.
D	Use the value of the StorHouse EDC system parameter, which sets the EDC value to the system default.

Refer to the StorHouse *System Administrator's Guide* for more information about EDC checking and all of the StorHouse system parameters that control EDC. Note how the following EDC system parameters affect StorHouse/RM:

EDC system parameters

EDC system parameter	When set to TRUE
EDC_INTERNAL	StorHouse performs an EDC check during query activity.
EDC_COPY	StorHouse performs an EDC check during file backup, archive, and write-back operations. A write-back occurs during a data load when VTF is NOW. In this case, when a FileTek data loader creates an extent, StorHouse does a write-back and checks EDCs. The load completes after all extents have been created and written back successfully.

File access group

A *file access group* provides a way to group files logically. For instance, you can assign files to be accessed by the same application or users to the same file access group. You can assign one group to each subspace. This means that table data, related indexes, and LOB data may be stored in the same or different groups. The default group name for StorHouse/RM is STH. Refer to the *StorHouse Concepts and Facilities Manual* and the StorHouse *System Administrator's Guide* for more information about file access groups and how to create them.

Maximum data extent size

You can set the maximum data extent size (in megabytes) for segment files. The range is 1 MB to the maximum surface size for the VSET. If you don't set the

maximum data extent size in a subspace, StorHouse/RM uses the following defaults:

Defaults for maximum data extent sizes

Default	Segment file type
Value of SQL_MAX_EXT_DATA system parameter	Table data files
Value of SQL_MAX_EXT_HASH system parameter	Hash index files
Value of SQL_MAX_EXT_VAL system parameter	Value index files
100 MB	LOB subsegment files

See Appendix C, “System parameters for StorHouse/RM,” for more information about these system parameters and recommendations on how to set them.

Extent retention in the performance buffer

StorHouse automatically migrates extents off the performance buffer based on factors such as extent size, ATF values, and access history. You can hold different types of extents in the performance buffer for a specified number of days (0 to 32,767days) to enhance access performance. Note, however, that StorHouse may migrate extents sooner if space is needed and no other data is available for migration.

For each subspace, you can assign an extent retention period for data extents and a different extent retention period for DF and map extents (also called *special*

extents). If you don't set extent retention periods for a subspace, StorHouse/RM uses the following defaults:

Defaults for extent retention periods

File type	Extent type	Default
Table data files	Data	Value of the SQL_HOLD_DATA system parameter
Index files	Data	Value of the SQL_HOLD_INDX system parameter
	Map	Value of the SQL_HOLD_SPECIAL system parameter
LOB subsegment files	Data	0 days
All files	DF	Value of the SQL_HOLD_SPECIAL system parameter

See Appendix C, "System parameters for StorHouse/RM," for more information about these system parameters and recommendations on how to set them.

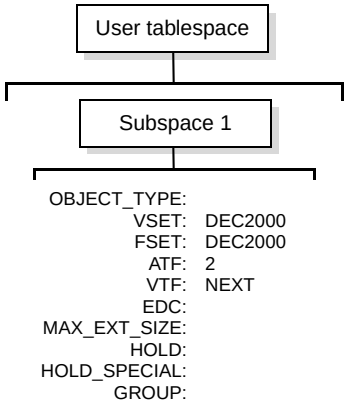
User tablespace examples

The number of user tablespaces you create or the number of subspaces for each user tablespace depends on your data and your access and performance requirements. Be sure to work with the StorHouse system administrator to define the best storage strategy for your data. This section contains examples of some of the ways you can define user tablespaces to meet different requirements.

Example 1: Minimal volume mounts

Assume you don't need to manage the storage of different data in different ways, and you want to minimize volume mounts.

Storage specifications. Create one user tablespace with one subspace for all component types and allocate one volume set and one file set.



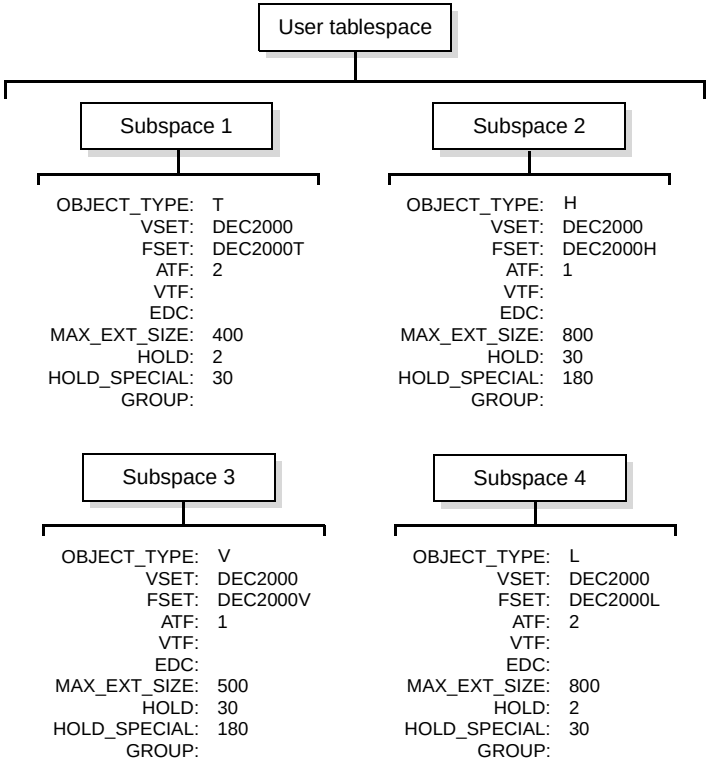
In this example, default values are used for all parameters except ATF and VTF.

Tablespace assignments. Assign the user table to the user tablespace and use the default user tablespace for indexes and LOB columns, that is, omit the TABLESPACE clause on the CREATE INDEX statements and LOB column definitions.

Example 2: Simultaneous removal

Assume you want to manage the storage of table data, indexes, and LOB data in different ways but remove them from StorHouse at the same time. Some reasons for simultaneous removal include the requirement to archive or back up table, index, and LOB data for off-site storage or to erase table, index, and LOB data with the same aging pattern from erasable volumes.

Storage specifications. Create one user tablespace with four subspaces, one subspace for each component type. Allocate four file sets in one volume set.

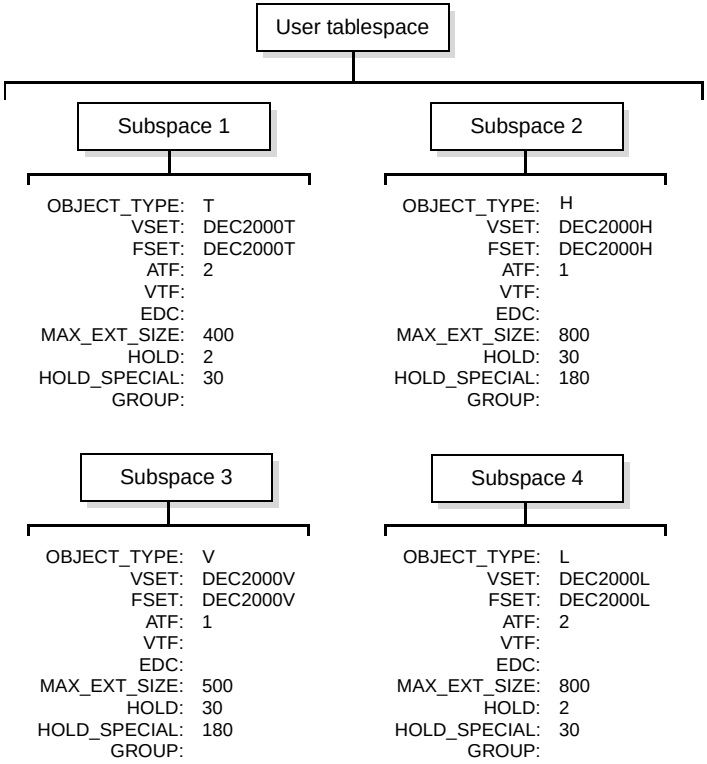


Tablespace assignments. Assign the user table, its indexes, and LOB columns to the same user tablespace.

Example 3: Load concurrency

Assume you want to manage the storage of table data, indexes, and LOB data in different ways but don't need to remove them from StorHouse at the same time. You also want to load table, index, and LOB data concurrently onto removable media.

Storage specifications. Create one user tablespace with four subspaces, one subspace for each component type. Allocate different volume sets and file sets for each subspace.



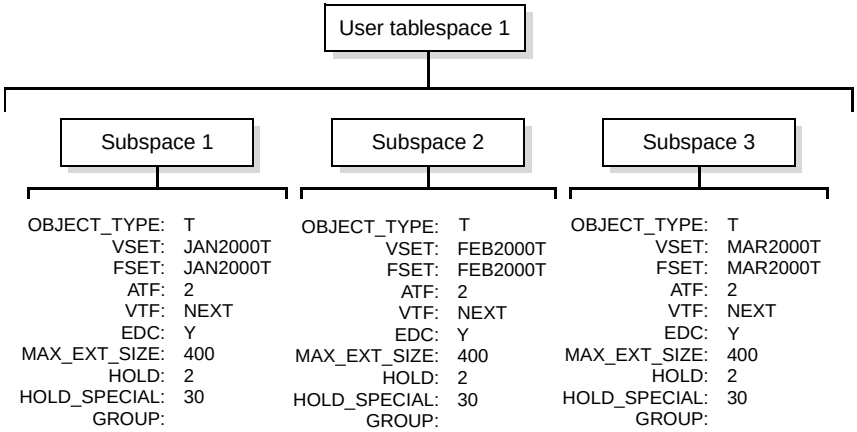
Tablespace assignments. Assign the user table, its indexes, and LOB columns to the same user tablespace.

Example 4: Access concurrency

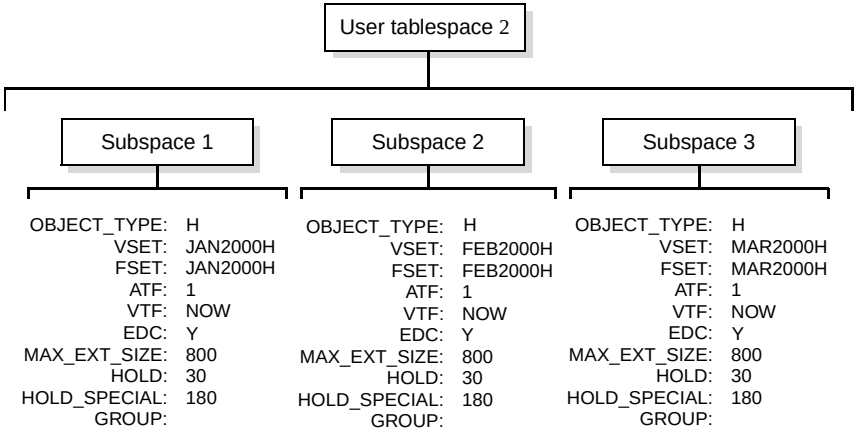
Assume you want to manage the storage of table data, indexes, and LOB data in different ways and don't need to remove them from StorHouse at the same time. You also plan to load data into the user table each month during a quarter and expect users to access multiple segments at a time.

Storage specifications. Create multiple user tablespaces, one for each component. (You can also do this with one user tablespace.) Define three subspaces in each user tablespace, one subspace for each month in a quarter. Allocate different volume sets and file sets for each subspace.

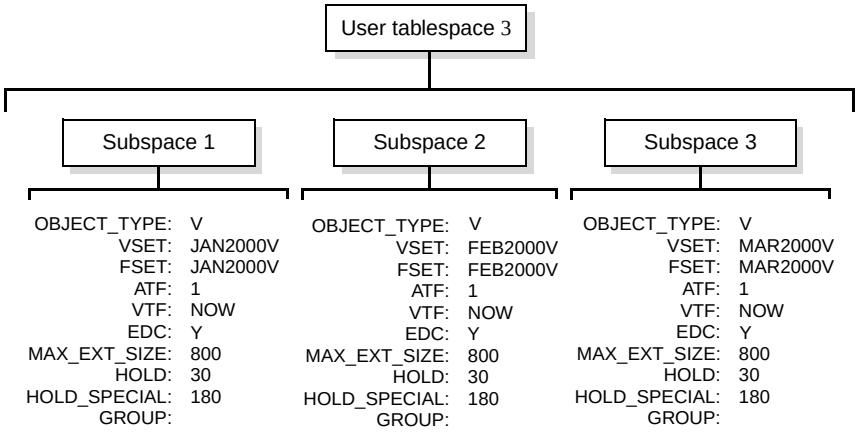
For instance, user tablespace 1 contains storage specifications for table data files only.



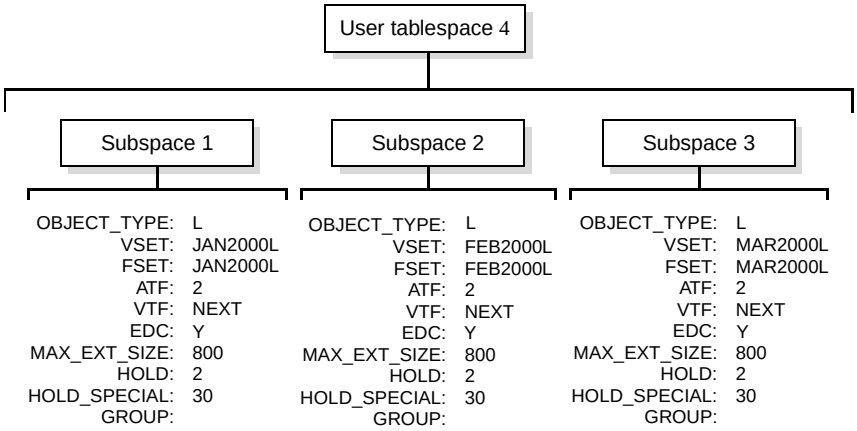
User tablespace 2 contains storage specifications for hash index files only.



User tablespace 3 contains storage specifications for value index files only.



User tablespace 4 contains storage specifications for LOB subsegment files only.



Note: To continue using the same user tablespaces and subspaces for the next quarter, you can create new volume sets and file sets and change the volume set and file set names in each subspace.

Tablespace assignments. Assign the user table to user tablespace 1, hash indexes to user tablespace 2, value indexes to user tablespace 3, and LOB columns to user tablespace 4.

Subspace selection during data load, index load, and segment merge operations

When loading table data, loading deferred indexes, and merging segments, a FileTek data loader determines the correct user tablespace(s) and automatically stores each segment file according to the appropriate subspace specifications. You can also explicitly select subspaces or rotate among eligible subspaces. Refer to the *FileTek FTP Data Loader Manual* or the *FileTek MVS Data Loader Utility Manual* for more information about subspace selection and rotation.

Default user tablespaces

A *default user tablespace* is the default that StorHouse/RM uses when you omit a TABLE SPACE clause on the CREATE TABLE and CREATE INDEX statements. You can assign a default user tablespace to a database and to each account. You make these assignments by inserting the account ID and user tablespace name into the SYSSMUSERS system table. To assign a default user tablespace to a database, use PUBLIC as the account ID.

Default user tablespace for a user table

When you create a user table and omit the TABLE SPACE clause, StorHouse/RM determines the default user tablespace as follows:

1. First, StorHouse/RM attempts to assign the user table to the account default user tablespace.

2. If there is no account default user tablespace, StorHouse/RM attempts to assign the user table to the database default user tablespace.

3. If there is no database default user tablespace, an error occurs.

Note: If you are creating a user table for another account, that is, you are the creator and another account is the owner, then StorHouse/RM uses *your* account default user tablespace if you omit the TABLE SPACE clause.

Default user tablespace for a LOB column

When you define a column definition for a LOB column and omit the optional TABLE SPACE clause, StorHouse/RM assigns the LOB column to the same user tablespace as the user table. In the following example, StorHouse/RM assigns the CLOB column named COLUMN2 to the default user tablespace and assigns the BLOB column named COLUMN3 to the BLOBSPACE user tablespace.

```
CREATE TABLE MYTABLE
(COLUMN1 INTEGER,
COLUMN2 CLOB,
COLUMN3 BLOB TABLESPACE BLOBSPACE)
TABLESPACE MYTABLESPACE
```

Default user tablespace for an index

When you create an index and omit the TABLE SPACE clause, StorHouse/RM assigns the index to the same user tablespace as the user table. In the following example, StorHouse/RM assigns the range index named PAY_DATE to the same user tablespace as the CUSTOMERS table.

```
CREATE RANGE INDEX PAY_DATE
ON CUSTOMERS (PDATE)
```

User tablespace naming conventions

Each user tablespace has a user tablespace name and a user tablespace ID. Each subspace has a subspace number.

User tablespace names

When you create a user tablespace, you give it a name. A user tablespace name must be unique in a database and should follow SQL identifier conventions. You must delimit user tablespace names that do not follow these conventions. Note that if you delimit a user tablespace name and that name contains lowercase characters (for instance, CREATE TABLE SPACE “May2000”), you must type the name in the same case (May2000, not MAY2000 or may2000) when inserting, updating, or deleting rows in SYSSMUSERS. If you do not delimit a user tablespace name, you must type it in uppercase characters on INSERT, UPDATE, or DELETE statements to SYSSMUSERS.

Tablespace IDs

StorHouse/RM assigns a *tablespace identifier* (ID) to each tablespace. This ID is unique within a database. Tablespace ID 0 identifies the system tablespace and tablespace ID 1 identifies the temporary tablespace. User tablespace IDs start with 2 and increment by 1 for each new user tablespace.

Subspace numbers

When you define a subspace, you assign a *subspace number*, which is a value from 0 to 2,147,483,647. You use subspace numbers when altering user tablespaces and when explicitly selecting subspaces during a load. Subspace numbers must be unique for each subspace in a user tablespace.

System tables with user tablespace information

The following system tables contain information about user tablespaces:

- SYSTBLSPACES – Contains the tablespace name and tablespace ID. StorHouse/RM inserts or deletes a row in this system table when you create or drop a user tablespace.
- SYSSTHSPACES – Contains storage specifications for each subspace in a user tablespace. StorHouse/RM inserts or updates one or more rows in this system table when you create or alter a user tablespace.
- SYSSMUSERS – Contains default user tablespace assignments for a database and accounts. You can maintain this system table, that is, insert, update, and delete rows to manage account and database default user tablespaces.

What's different about StorHouse user tablespaces?

The concept of tablespace differs by relational database management system, and how you work with tablespaces may also differ. For instance, in DB2 a tablespace is a physical VSAM dataset that contains table data. In StorHouse, a user tablespace is a logical component that defines storage specifications for table data, indexes, and LOB data.

Keep in mind the following when working with user tablespaces:

- A StorHouse database has one or more user tablespaces.
- Each user table, index, and LOB column belongs to a user tablespace.
- Indexes and LOB columns can belong to the same user tablespace as the table or to a different user tablespace(s).
- A user tablespace consists of one or more subspaces.
- A subspace can define storage specifications for all component types (table data, value index, hash index, or LOB data) or just one component type.
- You can alter a user tablespace to store subsequent segment files differently.
- You can assign a default user tablespace to each database and account.

Setting up a user tablespace

Setting up a new StorHouse user tablespace is a two-task process:

- Task 1 – Create volume sets and file sets for segment files with the StorHouse Command Language CREATE VSET and CREATE FSET commands.

- Task 2 – Create a user tablespace with the StorHouse SQL CREATE TABLE SPACE statement.

Creating volume sets and file sets

You can create volume sets and file sets for segment files with the StorHouse Command Language CREATE VSET and CREATE FSET commands. When you create volume sets and file sets, you name them. The following rules apply when choosing these names:

- Volume set and file set names can contain from 1 to 8 characters and consist of a–z, A–Z, 0–9, _ (underscore), and $ (dollar sign).
- A volume set name must be unique within StorHouse.
- A file set name must be unique within a volume set.
- Volume set and file set names are case insensitive. You cannot use case to distinguish volume set and file set names. For example, the volume set name MYVSET is the same as the volume set name myvset or MYvset.

If possible, choose a volume set or file set name that follows SQL identifier naming conventions. Use double quotes (“) to delimit volume set and file set names that start with a number or underscore (_), contain a $, or are an SQL reserved word. If you delimit a volume set or file set name, StorHouse/RM stores it in the SYSSTHSPACES system table in the same case as entered (lowercase, mixed case, uppercase); however, StorHouse always treats volume set and file set names as uppercase.

Refer to the StorHouse *Command Language Reference Manual* and the StorHouse *System Administrator’s Guide* for more information about the CREATE VSET and CREATE FSET commands.

▼ To create a volume set

Required privilege: ALLOCATION

1. Sign on to StorHouse.

2. At the command prompt (?), type the following and press `J` :

   CREATE VSET vset_name modifiers

   where vset_name is the name of the new volume set and modifiers are the desired command and parameter modifiers (to assign values other than the defaults). Some examples follow.

   – To create a volume set called JAN2000 using all of the command and parameter defaults, type:

     CREATE VSET JAN2000

   – To create a volume set called FEB2000H using all of the command and parameter defaults except /LIBRARY (to create the volume set on a library device other than the default), type:

     CREATE VSET FEB2000H /LIBRARY=L02

3. Record the volume set name because you need it when you create the user tablespace.

4. If needed, repeat this procedure to create a second volume set for the user tablespace.

▼ To create a file set

Required privilege: ALLOCATION

1. If necessary, sign on to StorHouse.

2. At the command prompt(?), type the following and press J :

   CREATE FSET fset_name modifiers

   where fset_name is the name of the new file set and modifiers are the desired command and parameter modifiers (to assign values other than the defaults). Below are some examples.

   – To create a file set called MY_TABLE for the volume set called JAN2000, using all of the command and parameter defaults, type:

     CREATE FSET MY_TABLE /VSET=JAN2000

   – To create a file set called FEB2000H for the volume set called FEB2000H, using all of the command and parameter defaults except /SIZE (to preallocate 500 megabytes of storage for the file set), type:

     CREATE FSET FEB2000H /VSET=FEB2000H /SIZE=500M

3. Record the file set name because you need it when you create the user tablespace.

4. If needed, repeat this procedure to create a second file set for the user tablespace.

Creating a user tablespace

You can create a user tablespace with the CREATE TABLE SPACE statement. (TABLE SPACE can be one or two words.)

Before you begin:

- Obtain the names of the volume set(s) and file set(s) that you or the StorHouse system administrator created for this user tablespace. Each subspace requires a volume set name and a file set name.

  Note: Remember to delimit volume set and file set names that do not follow SQL identifier conventions.

- Decide whether to use the default storage specifications. See "Displaying default storage specifications" on page 5-30 for information about displaying the current system default values.

- Choose a name for the user tablespace. See "User tablespace naming conventions" on page 5-20 for naming rules.

The following table summarizes the optional arguments on the CREATE TABLE SPACE statement. Refer to Chapter 4 in the *StorHouse SQL Reference Manual* for information about the CREATE TABLE SPACE statement.

Optional arguments on CREATE TABLE SPACE

Argument	Default	Values
OBJECT_TYPE type_value	blank	' ', T, H, V, L
ATF atf_value	0	0, 1, 2, 3
VTF vtf_value	DEFAULT	DEFAULT, NOW, NEXT, DIRECT
EDC edc_value	D	D, Y, N
GROUP group_name	STH	Valid StorHouse group
MAX_EXT_SIZE size_in_megabytes	0	1 to maximum surface size for the VSET
HOLD number_of_days	0	0 to 32767
HOLD_SPECIAL number_of_days	0	0 to 32767

▼ To create a user tablespace

Required privileges: SQLEXECUTE and DBA

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a CREATE TABLE SPACE statement for the user tablespace. Some examples follow.

   – To create a user tablespace called BILLINGTABLE with one subspace that uses all default values, type:

   ```
   CREATE TABLE SPACE BILLINGTABLE
   (SUBSPACE 1 VSET JAN2000T FSET JAN2000T)
   ```

   – To create a user tablespace called BILLINGJAN with four subspaces, one for each component, type:

   ```
   CREATE TABLE SPACE BILLINGJAN
   (SUBSPACE 1 VSET JAN2000T FSET JAN2000T OBJECT_TYPE T
   ATF 2 VTF NOW MAX_EXT_SIZE 400 HOLD 30
   HOLD_SPECIAL 180,
   SUBSPACE 2 VSET JAN2000H FSET JAN2000H OBJECT_TYPE H
   ATF 1 VTF NEXT MAX_EXT_SIZE 800 HOLD 90
   HOLD_SPECIAL 365,
   SUBSPACE 3 VSET JAN2000V FSET JAN2000V OBJECT_TYPE V
   ATF 1 VTF NEXT MAX_EXT_SIZE 500 HOLD 90
   HOLD_SPECIAL 365,
   SUBSPACE 4 VSET JAN2000L FSET JAN2000L OBJECT_TYPE L
   ATF 2 VTF NOW MAX_EXT_SIZE 800 HOLD 30
   HOLD_SPECIAL 180)
   ```

Assigning a default user tablespace

You can assign a default user tablespace to an account by inserting the account ID and user tablespace name into the SYSSMUSERS system table. You can assign a default user tablespace to a database by using PUBLIC as the account ID. StorHouse/RM uses a database default user tablespace when an account without an account default user tablespace omits the TABLE SPACE clause on the CREATE TABLE statement.

Caution: On the INSERT statement, type the account ID in uppercase. Also, if you delimited the tablespace name when you created the user tablespace, and the name contains lowercase letters (for instance, CREATE TABLE SPACE "Mar2000"), type the name in the same case (Mar2000, not MAR2000 or mar2000). If you didn't delimit the tablespace name, type it in uppercase letters.

▼ To assign a default user tablespace

Required privileges: SQLEXECUTE and either DBA or INSERT on SYSSMUSERS

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit an INSERT statement to insert the account ID and tablespace name into the SYSSMUSERS system table. Some examples follow.

   – To assign an account default user tablespace called mar2000 to USER1, type:

     ```
     INSERT INTO SYSADM.SYSSMUSERS (ACCOUNTID,
     DEFAULT_TS) VALUES ('USER1','mar2000')
     ```

   – To assign a database default user tablespace called DEFSPACE, type:

     ```
     INSERT INTO SYSADM.SYSSMUSERS (ACCOUNTID,
     DEFAULT_TS) VALUES ('PUBLIC','DEFSPACE')
     ```

Listing accounts and their default user tablespaces

You can list the account IDs and tablespace names of all accounts with default user tablespaces by submitting a SELECT statement on the SYSSMUSERS system table. The account ID PUBLIC identifies the database default user tablespace.

▼ To list all accounts and their default user tablespaces

Required privileges: SQLEXECUTE and either DBA or SELECT on SYSSMUSERS

1. Follow your site's procedure for connecting to your StorHouse database.
2. Submit the following SELECT statement on the SYSSMUSERS system table:

```
SELECT *
FROM SYSADM.SYSSMUSERS
```

Changing a default user tablespace

You can change the default user tablespace for a specific account or for a database by submitting an UPDATE statement on the SYSSMUSERS system table.

Caution: On the UPDATE statement, type the account ID in uppercase. Also, if you delimited the tablespace name when you created the user tablespace, and the name contains lowercase letters (for instance, CREATE TABLE SPACE "Mar2000"), be sure to type the name in the same case (Mar2000, not MAR2000 or mar2000). If you didn't delimit the tablespace name, type it in uppercase letters.

▼ To change a default user tablespace

Required privileges: SQLEXECUTE and either DBA or UPDATE on SYSSMUSERS

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit an UPDATE statement to change the default user tablespace. Some examples follow.

   – To change the account default user tablespace to apr2000 for USER1, type:

   ```
   UPDATE SYSADM.SYSSMUSERS
   SET DEFAULT_TS='apr2000'
   WHERE ACCOUNTID='USER1'
   ```

   – To change the database default user tablespace to DEFSPACE2, type:

   ```
   UPDATE SYSADM.SYSSMUSERS
   SET DEFAULT_TS='DEFSPACE2'
   WHERE ACCOUNTID='PUBLIC'
   ```

Displaying default storage specifications

You can display default storage specifications for all but OBJECT_TYPE and GROUP by submitting a StorHouse Command Language SHOW SYSTEM command. If you omit a parameter on the CREATE TABLE SPACE statement, StorHouse/RM uses the default. Note that the default value for OBJECT_TYPE is blank (for all component types) and the default value for GROUP is STH.

▼ To display default storage specifications

Required privilege: SHOW

1. Sign on to StorHouse.

2. At the command prompt (?), type the applicable SHOW SYSTEM command and press J .

– To display the default ATF value, type:

```
SHOW SYSTEM ATF
```

– To display the default VTF value, type:

```
SHOW SYSTEM VTF
```

– To display the default EDC value, type:

```
SHOW SYSTEM EDC
```

– To display the default MAX_EXT_SIZE value for a specific component, type one of the following:

For table data files – SHOW SYSTEM SQL_MAX_EXT_DATA
For hash index files – SHOW SYSTEM SQL_MAX_EXT_HASH
For value index files – SHOW SYSTEM SQL_MAX_EXT_VAL
For LOB subsegment files – The default MAX_EXT_SIZE value is 100 MB.

– To display the default HOLD value for a specific component, type one of the following:

For table data files – SHOW SYSTEM SQL_HOLD_DATA
For index files – SHOW SYSTEM SQL_HOLD_INDX
For LOB subsegment files – The default HOLD value is 0 days.

– To display the default HOLD_SPECIAL value, type:

```
SHOW SYSTEM SQL_HOLD_SPECIAL
```

Altering a user tablespace

You can change any of the storage specifications in a user tablespace with the ALTER TABLE SPACE statement. (TABLE SPACE can be one or two words.) Note the following:

- You cannot add or delete subspaces.
- When you change storage specifications, any new segment files are stored according to the new specifications. Existing segment files are stored according to the old specifications.

▼ To alter a user tablespace

Required privileges: SQLEXECUTE and DBA

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit an ALTER TABLE SPACE statement for the user tablespace. Some examples follow.

   – To change the volume set and file set names assigned to SUBSPACE 1 in the BILLINGTABLE user tablespace, type:

   ```
   ALTER TABLE SPACE BILLINGTABLE
   (SUBSPACE 1 VSET FEB2000T FSET FEB2000T)
   ```

   – To change the ATF value for SUBSPACE 1, the MAX_EXT_SIZE for SUBSPACE 2, and the number of days to hold special data extents in the performance buffer for SUBSPACE 3 in the BILINGJAN user tablespace, type:

   ```
   ALTER TABLE SPACE BILLINGJAN
   ( SUBSPACE 1 ATF 1 ,
   SUBSPACE 2 MAX_EXT_SIZE 400,
   SUBSPACE 3 HOLD_SPECIAL 65535 )
   ```

– To change the OBJECT_TYPE value for SUBSPACE 1 to allow all component types (no restriction), type:

```
ALTER TABLE SPACE BILLING2000
(SUBSPACE 1 OBJECT_TYPE ‘ ’)
```

Note that there must be a space between the quotes (after OBJECT_TYPE).

Listing user tablespace names and IDs

You can list a user tablespace ID, name, or both by submitting a SELECT statement on the SYSTBLSPACES system table. This procedure is useful if you are choosing a user tablespace name and need to determine whether it is unique.

▼ To list a user tablespace ID, or name, or both

Required privileges: SQLEXECUTE and either DBA or SELECT on SYSTBLSPACES

1. Follow your site’s procedure for connecting to your StorHouse database.

2. Submit a SELECT statement on the SYSTBLSPACES system table. Some examples follow.

   – To list all user tablespace IDs and names in a database, type:

   ```
   SELECT *
   FROM SYSADM.SYSTBLSPACES
   ```

   – To list all tablespace names in a database, type:

   ```
   SELECT TSNAME
   FROM SYSADM.SYSTBLSPACES
   ```

– To list all tablespace IDs in a database, type:

```
SELECT ID
FROM SYSADM.SYSTBLSPACES
```

– To list a tablespace ID for a specific user tablespace (such as BILLING2000), type:

```
SELECT ID
FROM SYSADM.SYSTBLSPACES
WHERE TSNAME='BILLING2000'
```

Displaying storage specifications for a user tablespace

You can display the storage specifications defined for each subspace in a user tablespace by submitting a SELECT statement on the SYSSTHSPACES system table. You need the tablespace ID to display the storage specifications for a specific user tablespace.

▼ To display storage specifications

Required privileges: SQLEXECUTE and either DBA or SELECT on SYSSTHSPACES

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a SELECT statement on the SYSSTHSPACES system table. Some examples follow:

   – To display all information about all user tablespaces in a database, type:

```
SELECT *
FROM SYSADM.SYSSTHSPACES
```

– To display the volume sets and file sets for all subspaces in a specific user tablespace with tablespace ID 3, type:

```
SELECT TSSUBSPACE_NUM,TSVSET,TSFSET
FROM SYSADM.SYSSTHSPACES
WHERE TSID='3'
```

– To display the ATF and VTF values for all subspaces in a specific user tablespace with tablespace ID 3, type:

```
SELECT TSSUBSPACE_NUM,TSATF,TSVTF
FROM SYSADM.SYSSTHSPACES
WHERE TSID='3'
```

Dropping a user tablespace

You can remove a user tablespace from the current database with the DROP TABLE SPACE statement. (TABLE SPACE can be one or two words.) Dropping a user tablespace removes all subspaces in that user tablespace.

Before you drop a user tablespace, you must drop all the user tables in that tablespace. When you drop the user tables, StorHouse/RM automatically drops the indexes associated with the user tables, even if the indexes are in a different user tablespace.

▼ To drop a user tablespace

Required privileges: SQLEXECUTE and DBA

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a DROP TABLE SPACE statement. The following example drops a user tablespace called BILLING2000:

   DROP TABLE SPACE BILLING2000

Chapter 6

User tables

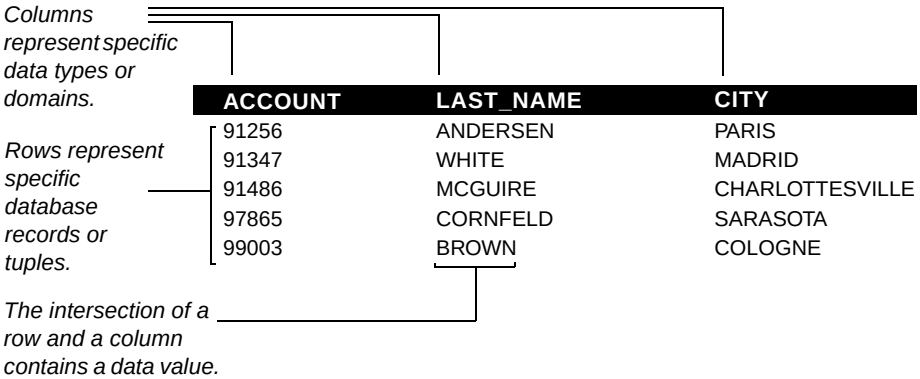
This chapter describes user tables and explains how to:

- Design a column of a user table
- Create a user table
- Display information about user tables
- List table names and table IDs
- Display a column definition for a user table
- Display volume set, file set, and segment information
- Drop a user table

About StorHouse user tables

A *user table* is a database component that holds user-accessible data. Logically, user tables consist of columns, rows, and data values. For instance, the following user table contains columns for customer account numbers, last names, and

cities. Each row contains the account number, last name, and city for a specific customer.



Physically, user table data is stored in table data files in segments. In-line LOBs are stored in the table data file, and out-of-line LOBs are stored in LOB subsegment files. See “Segmentation” on page 2-7 for more information about how table data and LOB data are stored.

Process of creating a user table

Before you can load data into a user table, you must define—or create—it. When you create a user table, you:

- Name the table. See “User table naming conventions” on page 6-3 for information about user table names.
- Define the columns of the table. See “Designing a column of a user table” on page 6-5 for more information about column definitions.
- Assign the user table to a user tablespace. See Chapter 5, “User tablespaces,” for more information about user tablespaces. If you don’t assign the user table to a user tablespace, StorHouse/RM assigns the table to a default user tablespace.

Owners and creators of user tables

Each user table has an owner. A *user table owner* has ALL database component privileges for the user table. These privileges include SELECT, INDEX, and INSERT (for loading only). In other words, the owner of a user table can retrieve data from the table, create indexes for the table, and load data into the table. In addition, a user table owner can grant privileges for that user table to other accounts.

Note: To load data into a user table, the owner also needs access and command privileges, as described on page 4-8.

Typically, the owner is the person who creates the user table, but accounts with DBA privilege can create user tables on behalf of other accounts. In this case, the person with DBA privilege is the *user table creator,* and the account for which they create the table is the user table owner. So, if account A has DBA privilege and creates a table for account B, account A is the creator and account B is the owner.

User table naming conventions

Each user table has a table name and a table ID.

User table names

When you create a user table, you give it a name. A table name should follow SQL identifier conventions. If a table name does not follow these conventions, you must delimit it and use the appropriate case in SQL statements.

A *fully qualified table name* is the combination of owner and table name. A table, view, and synonym name must be unique for an owner. For example, CALLSDBA.ACCOUNTS and SYSADM.ACCOUNTS are valid user table names in a database. A table called CALLSDBA.ACCOUNTS and a view called CALLSDBA.ACCOUNTS are not valid names because the combination is not unique.

Table IDs

StorHouse/RM assigns table IDs to user tables when you create them (and to system tables when you create a StorHouse database). Table IDs are unique within a database. The first user table ID in a database is 100. The first system table ID is 0.

System tables with user table information

Various system tables contain information about user tables.

- SYSTABLES – Contains one row for each user table (and system table and view) in a database. StorHouse/RM inserts a row into this system table when you create a user table or view and deletes a row when you drop a user table or view.

- SYSCOLUMNS – Contains one row for each column of every table in a database. For user tables, StorHouse/RM inserts a row into this system table for each column definition specified on the CREATE TABLE statement.

- SYSSTHFILES – Contains the StorHouse file names and group names of segment files. StorHouse/RM updates this system table after a confirmed data load operation, an index load operation, a segment replace and a merge operation, or when you drop a user table or index.

- SYSSTHSEGMENTS – Contains information about segments. StorHouse/RM updates this system table after a confirmed data load, a segment replace and a merge operation, or when you drop a user table.

What's different about StorHouse user tables?

Working with a user table in StorHouse may be different from working with one in your local database. In StorHouse:

- You cannot define primary or foreign keys for a user table because StorHouse does not support referential constraints.
- You cannot use an INSERT statement to add a row to a user table. Instead, you load rows into a user table with the FileTek MVS Data Loader utility or the FileTek FTP Data Loader.
- StorHouse does not support the UPDATE and DELETE statements on user tables. You cannot use an UPDATE statement to change a data value in a user table or a DELETE statement to remove a row from a user table.
- StorHouse does not support the ALTER TABLE statement on user tables. Once you create a user table, you cannot add a column, delete a column, or modify a data type in that table.

Designing a column of a user table

When creating a user table, you define the characteristics of each column. When designing a column, you:

- Decide the order of the columns
- Choose a name for the column
- Determine the data type of the column
- Decide whether the column can contain null values
- Specify a default value for the column, if desired
- Specify any LOB storage options

Deciding the order of columns

The order that you define columns affects the sequence in which columns appear in a result set when you issue SELECT * statement. For instance, if you define the following columns in this order—BILL_ACCOUNT, BILL_DATE, BILL_CATEGORY—then they appear in that order in the result set.

If a user table contains multiple LOB columns, StorHouse/RM determines in-line storage in column order. In other words, StorHouse/RM attempts to store the first LOB column value with the table data, and if space permits, attempts to store the second LOB column value with the table data, and so on. Therefore, the order you define LOB columns may affect which column values are stored with the table data. See page 6-12 for more information about LOB storage options.

Otherwise, column order does not affect performance. You can define columns that contain null values first or last, or you can define columns that contain variable-length data types first or last. Internally, StorHouse/RM stores column definitions to maximize performance.

Choosing a column name

Column names must be unique within a user table, but they do not have to be unique within a database. This means you can use the same column names in multiple user tables. For example, a BILLSUMMARY table can contain a column called BILL_ACCOUNT and a BILLDETAIL table can also contain a column called BILL_ACCOUNT. A column name should also follow SQL identifier conventions. You must delimit column names that do not follow these conventions.

Determining the data type

The *data type* of a column determines the maximum length of data values in the column and the kind of data that is valid for the column. By specifying a data

type you are enforcing an *integrity constraint* on a column. Prohibiting null values in a column is another integrity constraint (see page 6-9). When you load data into a user table or use an INSERT or UPDATE statement on a system table, the value that you provide for a column must be consistent with the length and data type of that column. StorHouse/RM truncates data strings longer than the allowable maximum for the data type.

StorHouse supports the following data types for defining columns. These are called *database data types*. For more information about database data types, refer to the *StorHouse SQL Reference Manual*. Note the following:

- The lengths are optional. StorHouse/RM uses a default length when you omit it.
- StorHouse/RM automatically compresses any VARCHAR and VARBINARY column with a length greater than 4096.

Data types for defining columns

Data type	Description	Fixed/ variable	Stored length
BINARY(length)	Array of bytes (8 bits each) with a length from 1 to 256 bytes	Fixed	Value of length
BLOB(length [K|M|G]) BINARY LARGE OBJECT(length [K|M|G])	Array of bytes with a length from ■ 1 to 2147483638 bytes ■ 1 to 2097152 K ■ 1 to 2048 M ■ 1 or 2 G	Variable	In-line: Size of BLOB plus 4 bytes Out-of-line: Size of BLOB in the LOB subsegment file and 22 bytes for the OID in the table data file

Data types for defining columns (continued)

Data type	Description	Fixed/ variable	Stored length
CHAR(length)	Array of characters with a length from 1 to 256 bytes	Fixed	Value of length
CLOB(length [K|M|G]) CHARACTER LARGE OBJECT(length [K|M|G])	Array of characters measured in bytes with a length from: ■ 1 to 2147483638 bytes ■ 1 to 2097152 K ■ 1 to 2048 M ■ 1 to 2 G	Variable	In-line: Size of CLOB plus 4 bytes Out-of-line: Size of CLOB in the LOB subsegment file plus 22 bytes for the OID in the table data file
DATE	Date value representing a month, day, and year	Fixed	4 bytes
DOUBLE PRECISION	Double precision, floating-point number	Fixed	8 bytes
INTEGER	Signed integer from -2147483648 through 2147483647, inclusive	Fixed	4 bytes
NUMERIC(P[,S]) DECIMAL(P[,S])	Decimal floating-point number with a precision (P) and a scale (S) up to 31 digits	Variable	(P/2)+1
REAL	Single precision, floating-point number	Fixed	4 bytes
SMALLINT	Small integer with values from -32768 through 32767, inclusive	Fixed	2 bytes

Data types for defining columns (continued)

Data type	Description	Fixed/ variable	Stored length
TIME	Time value representing an hour, minutes, seconds, and optionally milliseconds	Fixed	4 bytes
TIMESTAMP	Date and time combination representing a month, day, year, hour, minutes, seconds, and optionally milliseconds and microseconds	Fixed	8 bytes
VARBINARY(length)	Variable-length array of bytes (8 bits each) with a length from 1 to 32705 bytes (maximum length of 256 for indexed fields)	Variable	Actual size of data + 2 bytes
VARCHAR(length)	Variable-length array of characters with a length from 1 to 32705 bytes (maximum length of 256 for indexed fields)	Variable	Actual size of data + 2 bytes

Using null values in a column

StorHouse/RM has a value indicator, called a *null value*, to stand for an unknown, not applicable, or missing value. If a column allows null values, then you can load null values into that column.

For example, assume a column in a customer table contains middle initials. Not everyone has a middle initial, so you must be able to load data into the user table without requiring a middle initial for a customer record. In this case, you would allow null values in this column.

Caution: You cannot use the INSERT, UPDATE, or ALTER TABLE statements with user tables. If you define a column as null and load data without data values, then those data values will *always* be null. You cannot provide new or different values later.

It's important to note that a null value is not a data value in a column; rather, it's a null indicator. Each row contains a null bit for each column that allows nulls. This bit indicates whether null values are allowed. When you define a column, you can choose to allow or disallow null values. The default is to allow null values. You can disallow null values in two ways:

- Define a column as *not null*, which means you must provide a value for the column during a load operation.
- Define a default definition, which means that the column cannot be null and will have a default value if none is provided. See the next section, "Specifying defaults for a column," for more information about column defaults.

Defining a column as not null imposes an integrity constraint on the column, because not null means that you must *always* provide a data value for the column. StorHouse/RM rejects the request if you try to load, insert, or update data without data values for a column defined as not null and without a default definition.

Specifying defaults for a column

You can optionally specify a default value—called a *default definition*—for a column. A FileTek data loader inserts the default value into the column when there's no data value in the input data for that column.

Caution: You cannot use the INSERT, UPDATE, or ALTER TABLE statements with user tables. If you define a column with a default value and do not provide a specific data value, that data value will *always* be the default value. You cannot provide new or different data values later.

You can choose from a variety of default values, but some of your choices depend on the data type of the column, as described in the following table.

Column default values

Value	Data type	Description
Literal	Any	An integer, numeric string, or character string for columns of like data type. For example, for a column called STATE defined as CHAR(2), you could specify a default definition of VA. Data values in this column will contain the default value VA if you don't provide another value.
NULL	Any	A null value. This is the default if you omit a default definition.
SYSDATE	DATE	The current date.
SYSTIME	TIME	The current time.
USER	CHAR or VARCHAR	The account ID of the account performing the load. The column must have a minimum length of 12, for example, CHAR(12).

The default for a column is to allow null values. In other words, if you do not specify a default definition for a column, the column can be null.

Specifying LOB storage options

By default, a FileTek data loader attempts to store LOB data with the table data (in-line) in the same user tablespace as the table. For BLOB and CLOB column definitions, you can specify the following optional clauses:

LOB column definition options

To	Use this clause
Set the maximum length for in-line LOB data	INLINE [(length[K])]
Always store LOB data in a different file from the table data	NOT INLINE
Share storage with another LOB column	STORE WITH column_name
Use a different user tablespace from the table	TABLESPACE tablespace_name

To set the maximum length for in-line LOB data

You can set the maximum length for in-line LOB data by including the INLINE clause with a length (in K), for instance:

```
CREATE TABLE LOBTABLE
( CLOB_COLUMN CLOB INLINE(5K) )
TABLESPACE LOBSPACE
```

A FileTek data loader stores the LOB value in the row as long as the value does not exceed the maximum in-line length (5K for example) or the row does not exceed the maximum row length (32,705 bytes). If either length is exceeded, the data loader stores the value in a separate LOB subsegment file.

If you include the INLINE clause but omit the length, the default maximum length is 32,705 bytes. For instance, 32,705 bytes is the maximum length of the CLOB_COLUMN in the LOBTABLE in the following example:

```
CREATE TABLE LOBTABLE
( CLOB_COLUMN CLOB INLINE )
TABLESPACE LOBSPACE
```

If you omit both the INLINE clause and the NOT INLINE clause on a column definition, the default is to store the LOB data in-line as space permits. In other words, INLINE without a length specification is the default.

To store LOB values in a separate LOB subsegment file

You can store LOB column values in a separate file (out-of-line) from the table data by specifying the NOT INLINE clause on a column definition. In this case, the FileTek data loader stores an OID in the table data file and the LOB values in a LOB subsegment file. LOB values for a column are stored in the same LOB subsegment file; however, if a LOB subsegment file reaches the maximum size, (approximately 100 GB), the FileTek data loader creates another LOB subsegment file.

In the following example, the FileTek data loader stores CHAR_COLUMN and DATE_COLUMN values in the table data file, CLOB_COLUMN values in one LOB subsegment file, and BLOB_COLUMN values in another LOB subsegment file.

```
CREATE TABLE LOBTABLE2
( CHAR_COLUMN CHAR(32),
DATE_COLUMN DATE,
CLOB_COLUMN CLOB(1G) NOT INLINE,
BLOB_COLUMN BLOB NOT INLINE )
TABLESPACE LOBSPACE
```

To share storage with another LOB column

You can store the data of multiple LOB columns in the same LOB subsegment file by specifying the STORE WITH clause on a column definition. This enables multiple LOB values in multiple columns to share the same LOB subsegment file.

In the following example, the FileTek data loader stores CLOB_ONE values out-of-line with the CLOB_TWO values and attempts to store the BLOB_SMALL values in-line with the CHAR_COLUMN data.

```
CREATE TABLE LOBTABLE3
( CHAR_COLUMN CHAR(32),
CLOB_ONE CLOB NOT INLINE,
CLOB_TWO CLOB(1M) STORE WITH CLOB_ONE,
BLOB_SMALL BLOB(100K) )
TABLESPACE LOBSPACE
```

To store LOB values in a different user tablespace

You can assign a LOB column to a different user tablespace from the table by specifying the TABLESPACE clause on a column definition. If you omit this clause, the FileTek data loader assigns the LOB column to the same user tablespace as the table.

Note: For in-line LOBs, a FileTek data loader always uses the table's subspace and user tablespace, ignoring any LOB subspace or user tablespace.

In the following example, the FileTek data loader uses the CLOBSPACE user tablespace for the CLOB_ONE column and the LOBSPACE user tablespace for the PRIMARY_KEY, BLOB_ONE, and BLOB_TWO columns.

```
CREATE TABLE LOBTABLE4
( PRIMARY_KEY INTEGER,
BLOB_ONE BLOB INLINE(1K) ,
CLOB_ONE CLOB(1M) NOT INLINE TABLESPACE CLOBSPACE,
BLOB_TWO BLOB(100K) )
TABLESPACE LOBSPACE
```

Creating a user table

You can create a user table in a database by submitting a CREATE TABLE statement. You then load data into that user table with a FileTek data loader.

When you create a user table, you must provide:

- The table name
- At least one column definition

You can optionally provide:

- A default definition for each column
- The user tablespace name (FileTek recommends that you always provide this)

Note that if you don't provide:

- An owner name in the table name, then StorHouse/RM makes you the owner
- A default definition for a column, then a column can be null
- The user tablespace name, StorHouse/RM assigns a default user tablespace as described on page 5-19.

▼ To create a user table

Required privileges: SQLEXECUTE and either DBA or RESOURCE

1. Follow your site's procedure for connecting to your StorHouse database.
2. Submit a CREATE TABLE statement. Some examples follow.

– To create a four-column table called LOCATION for account ID USER1 in the ORDERS tablespace, type:

```
CREATE TABLE USER1.LOCATION
( LOC_NUM SMALLINT,
CITY_NAME CHAR(25),
STATE_ABBREV CHAR(2),
COUNTRY CHAR(25) )
TABLE SPACE ORDERS
```

– To create a four-column table called LOCATION for your account ID (SYSADM) in your default tablespace, type:

```
CREATE TABLE LOCATION
( LOC_NUM SMALLINT,
CITY_NAME CHAR(25),
STATE_ABBREV CHAR(2),
COUNTRY CHAR(25) )
```

– To create a four-column table called LOCATION for your account ID (SYSADM) in your default tablespace with a default definition of U.S.A. for the COUNTRY column, type:

```
CREATE TABLE LOCATION
( LOC_NUM SMALLINT,
CITY_NAME CHAR(25),
STATE_ABBREV CHAR(2),
COUNTRY CHAR(25) DEFAULT 'U.S.A.' )
```

– To create a table with a CLOB column using the default user tablespace, type:

```
CREATE TABLE LOBEXAMPLE1
( PRIMARY_KEY INTEGER,
CLOB_ONE CLOB )
```

In the preceding example, the FileTek data loader stores the CLOB_ONE column values in-line as long as the value or the row size does not exceed 32KB. If either exceeds the size limit, the data loader stores the PRIMARY_KEY values in the table data file and the CLOB_ONE values out-of-line in a LOB subsegment file in the default user tablespace.

– To create a table with multiple LOB columns using both in-line and out-of-line storage options, type:

```
CREATE TABLE LOBEXAMPLE2
( PRIMARY_KEY INTEGER,
CLOB_ONE CLOB INLINE(1K),
CLOB_SMALL CLOB(1M) NOT INLINE,
BLOB_SMALL BLOB(100K) )
TABLESPACE LOBSPACE
```

In the preceding example, the FileTek data loader stores CLOB_ONE values in-line for values up to 1K, BLOB_SMALL values in-line as long as they fit in the row, and CLOB_SMALL values out-of-line in a separate LOB subsegment file. If only one of the LOB values fits in the row, the data loader stores the CLOB_ONE values because that column is defined first.

– To create a table with multiple LOB columns that share storage, type:

```
CREATE TABLE LOBEXAMPLE3
( PRIMARY_KEY INTEGER,
CLOB_ONE CLOB,
CLOB_SMALL CLOB(1M) STORE WITH CLOB_ONE,
BLOB_SMALL BLOB(100K) STORE WITH CLOB_ONE )
TABLESPACE LOBSPACE
```

In the preceding example, the FileTek data loader stores the data in two files. The table data file contains the PRIMARY_KEY values plus any LOB values that fit in the table data row. The LOB subsegment file contains the CLOB_ONE, CLOB_SMALL, and BLOB_SMALL values that do not fit in the table row.

Displaying information about tables

You can display information about user and system tables in a database by submitting a SELECT statement on the SYSTABLES system table. This information includes the table creator and owner, the ID of the user tablespace to which the table is assigned, and the next segment ID to be loaded.

▼ To display information about tables

Required privileges: SQLEXECUTE and either DBA or SELECT on SYSTABLES

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a SELECT statement for the SYSTABLES system table. Some examples follow.

   – To display all information about all user and system tables in a database, type:

     ```
     SELECT *
     FROM SYSADM.SYSTABLES
     ```

   – To display all information about all tables owned by a specific owner, for instance, by SYSADM, type:

     ```
     SELECT *
     FROM SYSADM.SYSTABLES
     WHERE OWNER='SYSADM'
     ```

   – To display all information about a specific table (by table name), type:

     ```
     SELECT *
     FROM SYSADM.SYSTABLES
     WHERE TBL='LOCATION'
     ```

Listing table names and IDs

You can list a table name or ID by submitting a SELECT statement on the SYSTABLES system table. This task is helpful if you are choosing a user table name and want to determine whether the one you have chosen is unique for a database.

▼ To list a table ID or a table name

Required privileges: SQLEXECUTE and either DBA or SELECT on SYSTABLES

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a SELECT statement for the SYSTABLES system table. Some examples follow.

   – To display all table IDs and table names of all user and system tables in a database, type:

     ```
     SELECT ID, TBL
     FROM SYSADM.SYSTABLES
     ```

   – To display all table names for a specific owner, such as SYSADM, type:

     ```
     SELECT TBL
     FROM SYSADM.SYSTABLES
     WHERE OWNER='SYSADM'
     ```

Displaying a column definition

You can display a column definition for a user table by submitting a SELECT statement on the SYSCOLUMNS system table. This task is helpful if you want to

determine the data types of columns or whether a column allows null values or has default values. A column definition includes the following:

- Column name
- Data type
- Length of the data type
- Scale of a NUMERIC or DECIMAL data type
- Null indicator
- Default value, if specified
- LOB storage options

▼ To display a column definition for a user table

Required privileges: SQLEXECUTE and either DBA or SELECT on SYSCOLUMNS

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a SELECT statement on the SYSCOLUMNS system table. Some examples follow.

   – To display all column definitions for a specific table, such as the LOCATION table, type:

   ```
   SELECT *
   FROM SYSADM.SYSCOLUMNS
   WHERE TBL='LOCATION'
   ```

   – To display a specific column definition for a specific table, such as the STATE column in the LOCATION table, type:

   ```
   SELECT *
   FROM SYSADM.SYSCOLUMNS
   WHERE COL='STATE'
   AND TBL='LOCATION'
   ```

– To display all column definitions allowing null values, for all tables in a database, type:

```
SELECT *
FROM SYSADM.SYSCOLUMNS
WHERE NULLFLAG='Y'
```

– To list the table names with BLOB or CLOB columns and the in-line maximum length for each column, type:

```
SELECT TBL LOB_INLINE_MAX
FROM SYSADM.SYSCOLUMNS
WHERE COLTYPE IN ('BLOB' , 'CLOB')
```

Displaying volume set, file set, and segment information

You can display information about volume sets, file sets, and segment files by submitting StorHouse Command Language SHOW VSET, SHOW FSET, and SHOW FILE commands. This is useful if you want to display segment file sizes and volume set and file set properties.

Refer to the StorHouse *Command Language Reference Manual* for more information about these commands. See "Segment and LOB subsegment identifiers" on page 2-20 for more information about file name formats. The file name format differs by StorHouse/RM release.

▼ To display all available information about a volume set

Required privilege: SHOW

1. If necessary, sign on to StorHouse.

2. At the command prompt (?), submit the appropriate StorHouse Command Language SHOW VSET command and press J . An example follows.

   – To display all the available information about volume set JAN2000, type:

     SHOW VSET JAN2000 /FULL

     The system displays the following:

     VSET=JAN2000 DIRECTORY=PRIMARY SIZE=20459KB
     SURFACES=2 GENERAL_FREE=18604KB MEDIA=OEB LIMIT=0KB
     GENERAL_ALLOCATED=1854KB CREATED=01-JAN-2000:00:06:04
     MODIFIED=27-JAN-2000:14:03:16 CYCLE=0 DEACTIVATE=0
     EXPIRE=0 HOLD=NOHOLD LIBRARY=L00 ARCFSET=*
     BKPVSET=none BKPFSET=* MEMO="Rack 4"
     Total vsets displayed=1

▼ To display all available information about a file set

Required privilege: SHOW

1. If necessary, sign on to StorHouse.

2. At the command prompt (?), submit the appropriate StorHouse Command Language SHOW FSET command and press J . An example follows.

   – To display all available information about file set JAN2000 in volume set JAN2000, type:

     SHOW FSET JAN2000 /VSET=JAN2000 /FULL

The system displays the following:

```
FSET=JAN2000 VSET=JAN2000 DIRECTORY=PRIMARY
SIZE=1853KB GENERAL_FREE=2KB UPDATE_FREE=0KB
GENERAL_ALLOCATED=1165KB UPDATE_ALLOCATED=685KB
UPDATE_PERCENT=00 LIMIT=0KB CREATED=01-JAN-
2000:06:10:09

Total fsets displayed=1
```

▼ To display information about segment files

Required privilege: SHOW

1. If necessary, sign on to StorHouse.

2. At the command prompt (?), submit the appropriate StorHouse Command Language SHOW FILE command and press J . Some examples follow.

   – To display file name, group, version, and file ID information for all files in file set JAN2000 on volume set JAN2000, type:

     ```
     SHOW FILE * /VSET=JAN2000 /FSET=JAN2000
     ```

     The system displays the following:

     ```
     FILE="CALLS.00001.T0000000150.0000000000" GROUP=STH
     VERSION=0 FID=234.3

     Total files displayed = 1
     ```

   – To display file name, group, version, file ID, data, and size information for a table data file named CALLS.00001.T0000000150.0000000000, type:

     ```
     SHOW FILE CALLS.00001.T0000000150.0000000000 /BRIEF
     ```

The system displays the following:

```
FILE=“CALLS.00001.T0000000150.0000000000” GROUP=STH
VERSION=0 FID=234.3 DATE=22-JAN-2000:21:09:51 SIZE=8088

Total files displayed = 1
```

Note: By substituting /FULL for /BRIEF, you can display all available information about files, including the volume set and file set where they reside.

– To display extent information for a table data file named CALLS.00001.T0000000150.0000000000, type:

```
SHOW FILE CALLS.00001.T0000000150.0000000000
/EXTENT
```

The system displays all extents for the table data file, beginning with the most recently written extent. In this example, the DF extent is listed first (extent number 1000001), followed by the data extent (extent number 1000000).

```
FILE="CALLS.00001.T0000000150.0000000000" GROUP=STH
VERSION=0 FID=43726.636

EXTENT_NUMBER=1000001 EXTENT_SID=43726
EXTENT_DATE=22-JAN-2000:21:09:51
EXTENT_WRITTEN=22-JAN-2000:21:09:59 EXTENT_REVISION=1
EXTENT_SIZE=80 EXTENT_LOCATION=OEB“31F395EF”:A
EXTENT_LEVEL=L EXTENT_STATUS=(LAST) EXTENT_MF=none
EXTENT_RETENTION_DATE=25-JAN-2000:11:59:59

EXTENT_NUMBER=1000000 EXTENT_SID=43726
EXTENT_DATE=22-JAN-2000:21:09:51
EXTENT_WRITTEN=22-JAN-2000:21:09:53 EXTENT_REVISION=1
EXTENT_SIZE=800 EXTENT_LOCATION=OEB“31F395EF”:A
```

```
EXTENT_LEVEL=L EXTENT_STATUS=(none) EXTENT_MF=none
EXTENT_RETENTION_DATE=25-JAN-2000:11:59:59

EXTENTS_DISPLAYED=2

Total files displayed=1
```

Dropping a user table

You can drop a user table by submitting a DROP TABLE statement. When you drop a user table, StorHouse/RM:

- Removes the user table from the database
- Drops any indexes and privileges associated with the user table
- Marks the associated segment files for deletion

StorHouse/RM does not automatically drop any views dependent on the table, but it does invalidate them.

▼ To drop a user table

Required privileges: SQLEXECUTE and either DBA or table owner

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a DROP TABLE statement. Some examples follow.

   – To drop a table you own called LOCATION, type:

     ```
     DROP TABLE LOCATION
     ```

– To drop a table called CUSTOMER owned by account ID USER 1, if you have DBA privilege, type:

```
DROP TABLE USER1.CUSTOMER
```

Chapter 7

Indexes for user tables

This chapter describes indexes and explains how to:

- Create an index
- Display information about indexes
- List index names and IDs
- Display volume set, file set, and segment information
- Drop an index

About StorHouse indexes

User table *indexes* provide quick and efficient access to table data. StorHouse supports three types of indexes: value, hash, and range. You define the indexes, and a FileTek data loader creates the index entries during a load. You can display information about indexes or delete them when they are no longer needed.

The following general rules apply to indexes:

- You can create any index type for any column data type except BLOB and CLOB.

- You can create different types of indexes for the same column. For instance, you can create a range index and a value index for a column.

- The maximum length of a VARCHAR and VARBINARY column that you can index is 256.

- You must create a user table before you can create an index for it.
- The name of an index (combination of owner and index name) must be unique within a database.
- You can create an index of any type before or after you've loaded data into the table.
- A *deferred index* is an index created after a table has been loaded.
- You perform an *index load operation* to load a deferred index (create index entries for existing segments). Refer to the *FileTek FTP Data Loader Manual* and the *FileTek MVS Data Loader Utility Manual* for more information about loading deferred indexes.

Value indexes

A *value index* is a type of index based on a list (in ascending order) of all the values in a column (or group of columns for a compound value index). For each column value, a value index contains an index map to the table row containing that value. By searching the value index rather than the table, then matching column values to row IDs (also called *tuple IDs*), StorHouse/RM can more efficiently find a range of rows.

For example, the following value index on the BILL_ACCT column contains an ascending list of column values and their corresponding row IDs.

Value index

BILL_ACCT column	
Value	**Row ID**
10000	1
10001	2
20000	3
20001	4
30000	5

Queries that use value indexes

Value indexes work effectively with queries that return multiple rows based on a range of values. For instance, assume you created a value index on a column called BILL_ACCT in a user table called BILLING and then requested information for all account numbers between 10000 and 20000 with the following query:

```
SELECT BILL_ACCT, LAST_NAME, BILL_DATE
FROM BILLING
WHERE BILL_ACCT BETWEEN 10000 AND 20000
```

In this example, StorHouse/RM uses the value index on the BILL_ACCT column to locate the rows containing the range of values.

Value index

BILL_ACCT column	
Value	**Row ID**
10000	1
10001	2
20000	3
20001	4
30000	5

BILLING table

BILL_ACCT	LAST_NAME	BILL_DATE
10000	ANDERSEN	01/10/2000
10001	WHITE	01/02/2000
20000	MCGUIRE	01/12/2000
20001	CORNFELD	01/15/2000
30000	HAWKINS	01/01/2000

Value indexes and join processing

Value indexes can enhance join processing. For instance, a join may qualify for hybrid IN join processing when:

- The query is an equi-join
- The inner table in the join SQL has a value index on the join column

Refer to the *StorHouse SQL Reference Manual* for more information about joins, including hybrid IN.

How value indexes are stored

StorHouse/RM stores each value index as a *value index file* in a segment. For each value index on a table, there is always one value index file associated with each table data file. Each value index file is a separate STORHOUSE-type file. See "Index files" on page 2-9 for more information about value index files.

When you create a value index, you can assign it to a user tablespace. The value index files then are stored according to the specifications of the user tablespace. If you don't assign the index to a user tablespace, that is, you omit the TABLE SPACE clause on the CREATE INDEX statement, StorHouse/RM assigns the index to the same user tablespace as the user table. See Chapter 5, "User tablespaces," for more information about user tablespaces.

Hash indexes

A *hash index* is a two-part index based on an index map extent and a hit list that uses a proprietary StorHouse algorithm to effectively locate individual table rows based on individual index values. For hash indexes (unlike for value indexes), StorHouse/RM must access the data row to determine that the row's column content meets the selection criteria.

Queries that use hash indexes

Hash indexes work for queries that return specific rows based on a specific value. In other words, StorHouse/RM uses a hash index for queries that contain only equals-type predicates. StorHouse/RM uses a compound hash index for queries that contain equals-type predicates for all columns in the compound index.

For instance, StorHouse/RM might use a hash index for the following query because the query contains an equals-type predicate:

```
SELECT LAST_NAME, BILL_DATE
FROM BILLING
WHERE BILL_ACCT = 10000
```

In this example, you could create both a value index and a hash index on the BILL_ACCT column to satisfy both single row queries and queries that request a range of values. StorHouse/RM decides which index to use to satisfy a specific query.

How hash indexes are stored

StorHouse/RM stores each hash index as a *hash index file* in a segment. For each hash index on a table, there is always one hash index file associated with each table data file. Each hash index file is a separate STORHOUSE-type file. See "Index files" on page 2-9 for more information about hash index files.

When you create a hash index, you can assign it to a user tablespace. The hash index files then are stored according to the specifications of the user tablespace. If you don't assign the index to a user tablespace, that is, you omit the TABLE SPACE clause on the CREATE INDEX statement, StorHouse/RM assigns the index to the same user tablespace as the user table. See Chapter 5, "User tablespaces," for more information about user tablespaces.

Range indexes

For a specific column (or columns in the case of a compound index) within a user table, a *range index* contains the lowest and highest column data values for each segment. For instance, the following range index on the BILL_DATE column contains the low and high dates of a user table with five segments.

Range index

BILL_DATE column		
Segment	**Low Value**	**High Value**
1	01/01/1999	01/31/1999
2	02/01/1999	02/28/1999
3	03/01/1999	03/31/1999
4	04/01/1999	04/30/1999
5	05/01/1999	05/31/1999

Each time you load new data into a user table, StorHouse/RM enters the low and high data values into the range index along with the segment ID of the segment that contains them. If there's one segment, there's one low and high value pair in the range index. If there are 400 segments, there are 400 low and high value pairs in the range index.

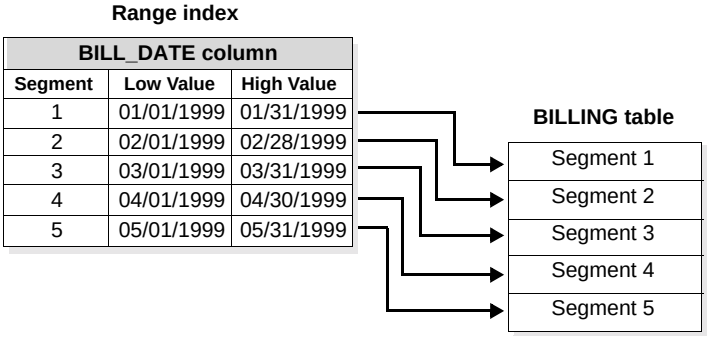
Queries that use range indexes

Range indexes are most effective for tables with multiple segments and when the data is partitioned on an indexed column. For instance, assume:

- Each month you load customer billing data into a table.
- Each row of data contains a billing date.
- The table contains a BILL_DATE column.

This data is partitioned by billing date because each segment contains a different billing date. By creating a range index for the BILL_DATE column, you can effectively search the table by date range and quickly find the segment containing the data for the billing month you want.

For example, assume the BILLING table consists of five segments and a range index is built on a BILL_DATE column. Segment 1 contains January data, segment 2 contains February data, and so on.



When responding to a query based on a specific date, StorHouse/RM can search the range index to determine which segment contains that date. It then searches the correct segment for the date, rather than searching sequentially through each segment.

For instance, if you issued the following query, StorHouse/RM uses the range index on the BILL_DATE column to quickly determine that segment 1 contains January data:

```
SELECT *
FROM BILLING
WHERE BILL_DATE='01/01/1999'
AND BILL_ACCT=30000
```

If the BILL_ACCT and BILL_DATE columns have hash indexes, StorHouse/RM can use those hash indexes to search for the specific account and date values in segment 1.

Range indexes and extractor processing

An *extraction* is a type of query that returns selected columns from all rows in a table. A query may qualify for extraction when:

- The query selects one or more entire segments
- All predicates in the WHERE clause are based on columns with range indexes
- The query meets all other extractor requirements

Refer to the *StorHouse ESQL Manual* for more information about extractor processing.

How range indexes are stored

StorHouse/RM stores range index entries in a set of system tables. There is one system table for each data type. When you load more data into a user table or when you load a deferred range index, StorHouse/RM updates the range index with the new segment's low and high data values for the indexed column or

columns. See “SYSRANGES” on page A-16 for more information about how range indexes are stored.

Simple and compound indexes

You can base an index on an individual column or on a combination of columns. An index based on one column is a *simple index*. An index based on multiple columns is a *compound* (or *composite*) index. You can create a compound index of any type, but as with simple indexes, you must indicate value, hash, or range when you create the index. For instance, you can create a compound value index called CUSTOMER based on the columns LAST_NAME, BILL_ACCT, and BILL_DATE in the BILLING table.

StorHouse/RM uses compound indexes when the comparison operator on all but the last column in a query is an equal sign (=) and all logical operators are AND. For example:

```
SELECT *
FROM BILLING
WHERE LAST_NAME=‘SMITH’ AND BILL_ACCT=‘1940339’ AND
BILL_DATE < ‘01/01/2000’
```

When you create a compound index, the column values are stored in the order you list them in the CREATE INDEX statement. For instance, if you base a compound index on DATE and TIME fields specifying DATE first, the index values are sorted on ascending times within ascending dates. If you specify TIME first, the index values are sorted on ascending dates within ascending times.

The following rules apply to compound indexes:

- A compound index can include columns of different data types.
- A compound index is either a range, value, or hash compound index.
- The maximum number of columns allowed in a compound index is 16.
- The maximum size of any column in a compound index is 256 bytes.

- The maximum size of a compound index is 4096 bytes. This means you can index 16 columns, each with a maximum size of 256 bytes.

Default index type

The StorHouse system parameter SQL_INDX_TYPE specifies the default index type—value, hash, or range—that StorHouse/RM creates when you omit the index type on a CREATE INDEX statement. You can change the default index type by changing the SQL_INDX_TYPE system parameter. See page C-8 for more information about the SQL_INDX_TYPE system parameter.

Index owners

The owner of a user table owns all of the indexes for that table, even if someone else creates the indexes. For instance, if you create a table for JACK, and then you create an index for that table, JACK owns the index, even though you created it.

Index naming conventions

Each index has an index name and an index ID.

Index names

When you create an index, you give it a name. An index name should follow SQL identifier conventions. If an index name does not follow these conventions, you must delimit it and use the appropriate case in SQL statements.

A *fully qualified index name* is the combination of owner name and index name. This combination must be unique from all other combinations in a database. In other words, more than one index can have the same descriptive index name as long as the owner names are unique. For example, JACK.BILL_DATE and

SANDY.BILL_DATE are valid index names within a database. Remember that the owner of the user table is automatically the owner of the index.

Index IDs

StorHouse/RM assigns index IDs to indexes when you create them. Index IDs are unique within a database. The first user table index ID in a database is 100, and the first system table index ID is 0.

System tables with index information

The following system tables contain information about indexes:

- SYSINDEXES – Contains one row for each indexed column of every table in a database. StorHouse/RM inserts a row (or multiple rows in the case of a compound index) into this system table when you create an index and deletes a row(s) when you drop an index. StorHouse/RM updates this system table after a deferred index load when all segments have indexes.

- SYSSTHFILES – Contains the StorHouse file names and group names of segment files, including index files. StorHouse/RM updates this system table after a confirmed data load, index load, segment replace and merge operation, or when you drop a user table or index.

What's different about StorHouse indexes?

Note the following considerations when working with indexes:

- StorHouse/RM does not support unique indexes for user tables.
- Index values are always stored in ascending order.

Creating an index for a user table

You can create a hash, value, or range index for one or more columns in a user table by submitting a CREATE INDEX statement. If you omit the TABLE SPACE clause, StorHouse/RM assigns the index to the same user tablespace as the table. If you omit the index type, StorHouse/RM uses the value of the SQL_INDX_TYPE system parameter. If the table contains data (has been loaded), you must create a deferred index.

▼ To create an index for a user table that hasn't been loaded

Required privileges: SQLEXECUTE and either DBA or RESOURCE or INDEX or table owner

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a CREATE INDEX statement for the column or columns you want to index. Some examples follow.

   – To create a simple value index called CUSTACCT on the ACCTNO column in the CUSTOMERS table, type:

     ```
     CREATE VALUE INDEX CUSTACCT
     ON CUSTOMERS (ACCTNO)
     ```

   – To create a compound (multi-column) hash index called CUSTACCTREP on the ACCTNO and SALESREP columns in the CUSTOMERS table, and to assign the hash index to the HASHINDEX user tablespace, type:

     ```
     CREATE HASH INDEX CUSTACCTREP
     ON CUSTOMERS (ACCTNO, SALESREP)
     TABLE SPACE HASHINDEX
     ```

– To create a simple range index called PAY_DATE on the PDATE column in the CUSTOMERS table, type:

```
CREATE RANGE INDEX PAY_DATE
ON CUSTOMERS (PDATE)
```

▼ To create a deferred index for a user table that has been loaded

Required privileges: SQLEXECUTE and either DBA or RESOURCE or INDEX or table owner

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a CREATE INDEX statement for the column or columns you want to index, using the DEFERRED keyword. For example, to create a range index called BILLING_END on the INVOICE_DATE column in the CUSTOMERS table, type:

```
CREATE RANGE INDEX BILLING_END
ON CUSTOMERS (INVOICE_DATE)
DEFERRED
```

Displaying information about indexes

You can display information about an index by submitting a SELECT statement on the SYSINDEXES system table. This information includes index owners, indexed columns, user table name and owner, index type (value, hash, range), and tablespace ID.

▼ To display information about indexes

Required privileges: SQLEXECUTE and either DBA or SELECT on SYSINDEXES

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a SELECT statement for the columns you want to display. Some examples follow.

   – To display the index ID of the CUSTACCT index owned by SMITH, type:

   ```
   SELECT ID
   FROM SYSADM.SYSINDEXES
   WHERE IDXNAME='CUSTACCT'
   AND IDXOWNER='SMITH'
   ```

   – To display the name of the column used for the CUSTACCT index for the ACCOUNTS table, type:

   ```
   SELECT COLNAME
   FROM SYSADM.SYSINDEXES
   WHERE IDXNAME='CUSTACCT'
   AND TBL='ACCOUNTS'
   ```

   – To display all the indexes defined for the ACCOUNTS table with their index types, type:

   ```
   SELECT IDXNAME, IDXTYPE
   FROM SYSADM.SYSINDEXES
   WHERE TBL='ACCOUNTS'
   ```

Listing index names and IDs

You can list the names and IDs of system and user table indexes by submitting a SELECT statement on the SYSINDEXES system table.

▼ To list index names and IDs

Required privileges: SQLEXECUTE and either DBA or SELECT on SYSINDEXES

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a SELECT statement on the SYSINDEXES system table. Some examples follow.

   – To list the names and IDs of all indexes in the database, type:

     ```
     SELECT IDXNAME, ID
     FROM SYSADM.SYSINDEXES
     ```

   – To list the names of all indexes on user tables and their associated user table names in the database, type:

     ```
     SELECT IDXNAME, TBL
     FROM SYSADM.SYSINDEXES
     WHERE TBL NOT LIKE 'SYS%'
     ```

     Because all system tables begin with the letters SYS, you are searching for all tables that do *not* begin with SYS, which are the user tables.

   – To list the names and owners of all indexes in a database, type:

     ```
     SELECT IDXNAME, IDXOWNER
     FROM SYSADM.SYSINDEXES
     ```

– To list the IDs and names of all indexes for CUSTOMERS table owned by SMITH, type:

```
SELECT ID, IDXNAME
FROM SYSADM.SYSINDEXES
WHERE TBL='CUSTOMERS'
AND TBLOWNER='SMITH'
```

Displaying volume set, file set, and segment information

You can display volume set, file set, and segment information for indexes by submitting StorHouse Command Language SHOW VSET, SHOW FSET, and SHOW FILE commands. This is useful if you want to display segment sizes, segment create dates, extent information, and volume set and file set properties.

Refer to the StorHouse *Command Language Reference Manual* for more information about these commands. See "Segment file names" on page 2-21 for more information about file name formats. Remember that the file name format differs by StorHouse/RM release.

▼ To display all available information about a volume set

Required privilege: SHOW

1. If necessary, sign on to StorHouse.

2. At the command prompt (?), submit the appropriate StorHouse Command Language SHOW VSET command and press J . An example follows.

– To display all the available information about volume set JAN1997, type:

```
SHOW VSET JAN1997 /FULL
```

The system displays the following:

```
VSET=JAN1997 DIRECTORY=PRIMARY SIZE=20459KB
SURFACES=2 GENERAL_FREE=18604KB MEDIA=OEB LIMIT=0KB
GENERAL_ALLOCATED=1854KB CREATED=01-JAN-1997:00:06:04
MODIFIED=29-JAN-1997:19:07:14 CYCLE=0 DEACTIVATE=0
EXPIRE=0 HOLD=NOHOLD LIBRARY=L00 ARCFSET=*
BKPVSET=none BKPFSET=* MEMO="Rack 4"

Total vsets displayed=1
```

▼ To display all available information about a file set

Required privilege: SHOW

1. If necessary, sign on to StorHouse.

2. At the command prompt (?), submit the appropriate StorHouse Command Language SHOW FSET command and press J . An example follows.

   – To display all available information about file set JAN1997 in volume set JAN1997, type:

```
SHOW FSET JAN1997 /VSET=JAN1997 /FULL
```

The system displays the following:

```
FSET=JAN1997 VSET=JAN1997 DIRECTORY=PRIMARY
SIZE=1853KB GENERAL_FREE=2KB UPDATE_FREE=0KB
GENERAL_ALLOCATED=1165KB UPDATE_ALLOCATED=685KB
UPDATE_PERCENT=00 LIMIT=0KB CREATED=01-JAN-
1997:06:10:09 MODIFIED=29-JAN-1997:19:07:14
```

```
Total fsets displayed=1
```

▼ To display information about an index file

Required privilege: SHOW

1. If necessary, sign on to StorHouse.

2. At the command prompt (?), submit the appropriate StorHouse Command Language SHOW FILE command and press J . Some examples follow.

   – To display file name, group, version, and file ID information for all files on file set JAN1997 on volume set JAN1997 (the * is a wildcard character), type:

     ```
     SHOW FILE * /VSET=JAN1997 /FSET=JAN1997
     ```

     The system displays the following (note that the file names are in 2.1 to 2.2A format):

     ```
     FILE="CUSTOMERS.HQSM.000000130.H.0001" GROUP=STH
     VERSION=0 FID=234.2

     FILE="CUSTOMERS.HQSM.000000130.H.0002" GROUP=STH
     VERSION=0 FID=234.333

     Total files displayed = 2
     ```

   – To display file name, group, version, file ID, data, and size information for all index files named CUSTOMERS.HQSM.000000130.H.* on StorHouse, type:

     ```
     SHOW FILE CUSTOMERS.HQSM.000000130.H.* /BRIEF
     ```

The system displays the following:

```
FILE=“CUSTOMERS.HQSM.000000130.H.0001” GROUP=STH
VERSION=0 FID=234.2 DATE=22-JAN-1997:21:09:51 SIZE=7488

FILE=“CUSTOMERS.HQSM.000000130.H.0002” GROUP=STH
VERSION=0 FID=234.333 DATE=29-JAN-1997:19:03:25 SIZE=398

Total files displayed = 2
```

Note: By substituting /FULL for /BRIEF, you can display all available information about the files, including the volume set and file set where they reside.

– To display extent information for a value index file named STORHSE1ACCTNG97.SYSTEMA.000000161.V.0000, type:

```
SHOW FILE STORHSE1ACCTNG97.SYSTEMA.000000161.V.0000
/EXTENT
```

The system displays all extents, beginning with the most recently written extent. In this example, the extents are listed in this order: map extent (number 1000002), DF extent (number 1000001), and data extent (number 1000000).

```
FILE="STORHSE1ACCTNG97.SYSTEMA.000000161.V.0000"
GROUP=STH VERSION=0 FID=43726.225

EXTENT_NUMBER=1000002 EXTENT_SID=43726
EXTENT_DATE=21-JAN-1997:14:02:47
EXTENT_WRITTEN=21-JAN-1997:14:02:47 EXTENT_REVISION=1
EXTENT_SIZE=88 EXTENT_LOCATION=OEB“312C89B7”:A
EXTENT_LEVEL=L EXTENT_STATUS=(BUFFERED,LAST,NEW)
EXTENT_MF=800000 EXTENT_RETENTION_DATE=21-JAN-
1999:11:59:59
```

```
EXTENT_NUMBER=1000001 EXTENT_SID=43726
EXTENT_DATE=21-JAN-1997:14:02:43
EXTENT_WRITTEN=21-JAN-1997:14:02:44 EXTENT_REVISION=1
EXTENT_SIZE=80 EXTENT_LOCATION=OEB“312C89B7”:A
EXTENT_LEVEL=L EXTENT_STATUS=(BUFFERED,LAST,NEW)
EXTENT_MF=400000 EXTENT_RETENTION_DATE=21-JAN-
1999:11:59:59

EXTENT_NUMBER=1000000 EXTENT_SID=43726
EXTENT_DATE=21-JAN-1997:14:02:42
EXTENT_WRITTEN=21-JAN-1997:14:02:42 EXTENT_REVISION=1
EXTENT_SIZE=750 EXTENT_LOCATION=OEB“312C89B7”:A
EXTENT_LEVEL=L EXTENT_STATUS=(BUFFERED,NEW)
EXTENT_MF=100000 EXTENT_RETENTION_DATE=21-JAN-
1999:11:59:59

EXTENTS_DISPLAYED=3

Total files displayed=1
```

Dropping an index

You can drop an index on a user table by submitting a DROP INDEX statement. If the index is a value or hash index, this statement removes all index files (StorHouse primary files) for all segments in the user table. If the index is a range index, this statement removes all of the index entries from the range index system tables.

▼ To drop an index on a user table

Required privileges: SQLEXECUTE and either DBA (to drop an index owned by another account) or RESOURCE or INDEX or table owner

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a DROP INDEX statement for the index you want to delete. For example, to remove the CUSTINDEX index on the CUSTOMER table, type:

   ```
   DROP INDEX CUSTINDEX ON CUSTOMER
   ```

Chapter 8

Views

This chapter describes views and explains how to:

- Create a view
- Display information about views
- List view names
- Drop a view

About views

A *view* is a temporary database component that draws its contents from one or more existing tables and/or views. When you query a view, you actually query the tables referenced by the view.

Views tailor or limit access to information. They provide convenience, security, or both by letting you determine who can access what data. For convenience, you might construct a two-column view that derives customer names from the CUSTOMER table and their associated sales representatives from the SALESREPS table. For security reasons, you might construct a view that excludes commission information from SALESREPS.

How views work

You create views by pairing the CREATE VIEW statement with a SELECT statement. CREATE VIEW names the view. The SELECT statement contains a

query that defines the view. For example, suppose you have the following four-column table called SALESREPS:

SALESREPS table

SALES_REP	EMP_NUM	CITY	COMMISSION
ANDERSEN	0001000	PARIS	.03
WHITE	0002000	MADRID	.07
HOHMAN	0003000	COLOGNE	.03
REID	0004000	PHILADELPHIA	.05
MCGUIRE	0005000	CHARLOTTESVILLE	.04
SHOVLIN	0006000	PHILADELPHIA	.10
MCKENNA	0007000	LONDON	.09
CORNFELD	0008000	SARASOTA	.02

To create a view that displays only those sales representatives who are assigned to Philadelphia, you would issue the following SQL statement:

```
CREATE VIEW PHILLYSTAFF
AS SELECT *
FROM SALESREPS
WHERE CITY = ‘PHILADELPHIA’
```

Whenever you access the PHILLYSTAFF view, StorHouse/RM executes the associated SELECT statement on the SALESREPS table to find those table entries that contain PHILADELPHIA in the CITY column.

For instance, if you issue this query to display all information in the PHILLYSTAFF view:

```
SELECT *
FROM PHILLYSTAFF
```

In response, StorHouse/RM executes the following SQL:

```
SELECT *
FROM SALESREPS
WHERE CITY = 'PHILADELPHIA'
```

The result of this query is:

PHILLYSTAFF View

SALES_REP	EMP_NUM	CITY	COMMISSION
REID	0004000	PHILADELPHIA	.05
SHOVLIN	0006000	PHILADELPHIA	.10

In summary, the PHILLYSTAFF view is a stored query that accesses the SALESREPS table whenever you access the view. When information in SALESREPS changes, the PHILLYSTAFF view may also change because the stored query may return different results. For example, if you load more data into the SALESREPS table, the next time you query the PHILLYSTAFF view, any new sales representatives are included in the results.

Owners and creators of views

Each view has an owner. A view *owner* has ALL table privileges for the view. (This does not include the INDEX privilege because you cannot add an index to a view.) In addition, a view owner can grant privileges for that view to other accounts. Typically, an owner is the account that creates the view, but accounts with DBA privilege can create views on behalf of other accounts by specifying an owner name in the CREATE VIEW statement. In this case, the account with DBA privilege is the view *creator*, and the account with the specified owner name is the view owner.

For example, in the following statement, account SSC is the owner of the view LONDON_STAFF:

```
CREATE VIEW SSC.LONDON_STAFF
AS SELECT *
FROM SALESREPS
WHERE CITY = 'LONDON'
```

View naming conventions

When you create a view, you give it a name. A view name should follow SQL identifier conventions. If a view name does not follow these conventions, you must delimit it and use the appropriate case in SQL statements.

A *fully qualified view name* is the combination of owner name and view name. A view, table, and synonym name must be unique for an owner. For example, CALLSDBA.OCTBILL and SYSADM.OCTBILL are valid view names in a database. A view called CALLSDBA.OCTBILL and a synonym called CALLSDBA.OCTBILL are not valid names because the combination of owner name and view name is not unique. If you omit the owner name when creating a view, then StorHouse/RM makes you the owner.

View restrictions

The following restrictions apply to views:

- You must base a view on a single query. Subqueries are not allowed.
- The query may not contain an ORDER BY clause.
- You cannot use DELETE, INSERT, or UPDATE statements with views derived from user tables. These statements are restricted to system tables and views based on system tables.

View privileges

The following rules apply:

- To create a view, you must have DBA or RESOURCE privilege.
- If you have RESOURCE privilege, you must also have SELECT privilege on any tables that the view references.
- When you create a view, you acquire a set of default privileges. These may include privileges you don't have on the underlying tables. However, when you attempt to access the view, StorHouse/RM will not let you perform operations on the view that you're not allowed to perform on the underlying tables.
- If your privileges on underlying tables were issued using the WITH GRANT OPTION of the GRANT command, you can grant privileges on your view to other accounts.
- When you grant privileges on the view to other accounts, the other accounts need not have privileges on the underlying tables.

System tables with view information

The following system tables contain information about views:

- SYSTABLES – Contains one row for each view (and system table and user table) in a database. StorHouse/RM inserts or deletes rows in this system table when you create or drop views (and user tables).
- SYSVIEWS – Contains one row for each view in a database. StorHouse/RM inserts or deletes rows in this system table when you create or drop views.

- SYSTABAUTH – Contains account privileges for views (and tables). StorHouse/RM maintains this system table when you grant and revoke privileges.

Creating a view

You can create a view by submitting a CREATE VIEW statement. This statement contains the view name and the query constructs the view.

Note the following:

- You can create views from tables, views, or both. The procedure is the same in every case.
- By default, the column names of a view are the same as the column names of underlying tables unless you explicitly rename them when creating the view.

▼ To create a view

Required privileges: SQLEXECUTE and either DBA or RESOURCE and SELECT on referenced tables

1. Follow your site's procedure for connecting to your StorHouse database.
2. Submit a CREATE VIEW statement for the view you want to create. For example, to create a view named PHILLYSTAFF that only displays the Philadelphia sales representatives from the SALESREPS table, type:

```
CREATE VIEW PHILLYSTAFF
AS SELECT *
FROM SALESREPS
WHERE CITY = 'PHILADELPHIA'
```

Creating views from multiple tables

One of the most useful features of views is that you can combine tables, tables and views, or views and views to create useful virtual tables containing information that you could not get from any of the tables or views alone.

For instance, suppose that in addition to the SALESREPS table that matches representatives with cities and commission rates, you also have a SALARY table that includes everyone in your company, their positions, and their annual salaries.

SALARY table

EMPLOYEE	EMP_NUM	POSITION	SALARY
ANDERSEN	0001000	SALES_REP	$35,500
CORNFELD	0008000	SALES_REP	$19,500
MCGUIRE	0005000	SALES_REP	$42,000
HEMINGWAY	0011000	COPY_WRITER	$55,000
HOLT	0003000	SALES_REP	$480,000
MATISSE	0010000	DESIGNER	$36,000
MCKENNA	0007000	SALES_REP	$87,000
REID	0004000	SALES_REP	$30,500
SHOVLIN	0006000	SALES_REP	$36,000
WHITE	0002000	SALES_REP	$235,000
WILKINS	0009000	ACCOUNTANT	$75,000

To create a view called SALES_PAY displaying all sales representatives, their commission rates, and their annual salaries, you could issue the following SQL statement to combine this information from the SALESREPS and SALARY tables:

```
CREATE VIEW SALES_PAY
AS SELECT SALES_REP, SALESREP.EMP_NUM, COMMISSION, SALARY
FROM SALESREPS, SALARY
WHERE POSITION = 'SALES_REP'
AND SALES_REP = EMPLOYEE
```

Notice that within the query, you must precede the column name EMP_NUM with the table name SALESREPS. This is because EMP_NUM is a column name in both the SALESREPS and SALARY tables. Whenever such an ambiguity occurs, you must resolve it by providing the fully qualified name of the table or view that you intend to use.

To see all rows of the SALES_PAY view, issue the query:

```
SELECT *
FROM SALES_PAY
```

The query produces the following:

SALES_PAY View

SALES_REP	EMP_NUM	COMMISSION	SALARY
ANDERSEN	0001000	.03	$35,500
WHITE	0002000	.07	$235,000
REID	0004000	.05	$30,500
MCGUIRE	0005000	.04	$42,000
SHOVLIN	0006000	.10	$36,000
MCKENNA	0007000	.09	$87,000
CORNFELD	0008000	.02	$19,500
HOLT	0003000	.03	$480,000

▼ To create a view based on multiple tables (or views)

Required privileges: SQLEXECUTE, DBA or RESOURCE, and SELECT on referenced tables

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a CREATE VIEW statement for the view you want to create. For example, to create a view named SALES_PAY that combines the SALESREPS and SALARY tables to display all sales representatives, their commission rates, and their annual salaries, type:

```
CREATE VIEW SALES_PAY
AS SELECT SALES_REP, SALESREP.EMP_NUM, COMMISSION,
SALARY
FROM SALESREPS, SALARY
WHERE POSITION = 'SALES_REP'
AND SALES_REP = EMPLOYEE
```

Naming columns in views

By default, when you create a view, the names of the columns in the view are the same as those in the base tables. However, there may be times when you want to rename the columns. For instance, you must rename columns when creating a view from two tables with columns that have the same name.

You can rename columns in the CREATE VIEW statement by following the name of the view with new column names in parentheses. For instance, you could have created the SALES_PAY view to display the output in columns labeled SALES_PERSONS, NUM, COM, and SALARY as shown in the following example.

Although you don't intend to change every column name, you must specify all four column identifications. This is because if you name one output column, you must name all output columns.

▼ To create a view with column names that differ from the column names in the base tables

Required privileges: SQLEXECUTE, DBA or RESOURCE, and SELECT on referenced tables

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a CREATE VIEW statement name that includes new column names in parentheses for the view you want to create. For example, to create a view with columns named SALES_PERSONS, NUM, COM, AND SALARY, type:

```
CREATE VIEW SALES_PAY (SALES_PERSONS,NUM,COM,SALARY)
AS SELECT SALES_REP, SALESREP.EMP_NUM, COMMISSION,
SALARY
FROM SALESREPS, SALARY
WHERE POSITION = 'SALES_REP'
AND SALES_REP = EMPLOYEE
```

Displaying information about views

You can display information about views by submitting a SELECT statement on the SYSVIEWS system table. View information includes view name, view owner and creator, and the query that constructs the views.

▼ To display information about views

Required privileges: SQLEXECUTE and DBA or SELECT on SYSVIEWS

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a SELECT statement on the SYSVIEWS system table. Some examples follow.

– To list all views in a database with the SQL statements that define them, type:

```
SELECT *
FROM SYSADM.SYSVIEWS
```

– To display information about all views owned by a specific account, for instance, by SYSADM, type:

```
SELECT *
FROM SYSADM.SYSVIEWS
WHERE OWNER='SYSADM'
```

– To display the SQL definition of a specific view (by view name), for instance, SALESREPS, type:

```
SELECT VIEWTEXT
FROM SYSADM.SYSVIEWS
WHERE VIEWNAME='SALESREPS'
```

Listing view names

You can list the view names in a database by submitting a SELECT statement on the SYSVIEWS system table.

▼ To list view names

Required privileges: SQLEXECUTE and DBA or SELECT on SYSVIEWS

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit the following SELECT statement on the SYSVIEWS system table.

```
SELECT VIEWNAME,OWNER
FROM SYSADM.SYSVIEWS
```

Dropping a view

You can drop a view by submitting a DROP VIEW statement. Views that reference a dropped view become invalid but are not automatically removed from StorHouse.

▼ To drop a view

Required privileges: SQLEXECUTE and DBA (to drop a view owned by another account) or view owner.

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a DROP VIEW statement for the view you want to drop. Some examples follow.

   – To drop a view named SALES_SAL that you own, type:

   ```
   DROP VIEW SALES_SAL
   ```

   – To drop a view named CORP_SALES owned by account USER1, type:

   ```
   DROP VIEW USER1.CORP_SALES
   ```

Chapter 9

Synonyms

This chapter describes synonyms and explains how to:

- Create a synonym
- Rename a synonym
- Display information about synonyms
- List synonym names
- Drop a synonym

About synonyms

A *synonym* is another name for a table, view, or synonym. You can create a synonym and then use it instead of the original component name in the following SQL statements:

- SELECT
- GRANT
- REVOKE
- INSERT (for system tables and loading user tables)
- UPDATE (for system tables only)
- DELETE (for system tables only)

Synonyms are convenient because you don't have to use fully qualified table names. For instance, you can submit this query with the synonym BILLS:

```
SELECT BILL_ACCT,BILL_AMOUNT
FROM BILLS
```

instead of this query with the fully qualified table name:

```
SELECT BILLACCT,BILL_AMOUNT
FROM CALLSDBA.BILLSUMMARY
```

Private and public synonyms

A synonym may be private or public. A *private synonym* belongs to the account who creates it. Synonyms are private by default. To use a private synonym, an account must meet one of the following requirements:

- Own the synonym
- Have DBA privilege
- Have SELECT privilege on the component the synonym represents

A *public synonym* belongs to PUBLIC, which means any StorHouse account can use the synonym. Only accounts with DBA privilege can create and drop public synonyms.

Synonym owners

The account who creates the synonym is also the synonym owner. A synonym owner can drop the synonym.

Synonym naming conventions

When you create a synonym, you give it a name. A synonym name should follow SQL identifier conventions. If a synonym name does not follow these conventions, you must delimit it and use the appropriate case in SQL statements.

A *fully qualified synonym name* is the combination of owner name and synonym name. A table, view, and synonym name must be unique for an owner. For

instance, a synonym called SYSADM.BILLS and a table called CALLSDBA.BILLS are valid names in a database. A synonym called SYSADM.BILLS and a view called SYSADM.BILLS are not unique and therefore not valid names.

System table with synonym information

The SYSSYNONYMS system table contains the name of each synonym, its owner and creator, and the name and owner of the database component the synonym represents. StorHouse/RM inserts and deletes rows in this system table when you create and drop synonyms.

Creating a synonym

You can create a private or public synonym with the CREATE SYNONYM statement. Synonyms are private by default. When you create a synonym, you specify the name of the synonym and the name of the database component the synonym represents.

▼ To create a private synonym

Required privileges: SQLEXECUTE and DBA or RESOURCE

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a CREATE SYNONYM statement for the table, view, or synonym you want to represent with a synonym. For example, to create a private synonym called NEW for the table SYSADM.NEWEMPLOY, type:

```
CREATE SYNONYM NEW
FOR SYSADM.NEWEMPLOY
```

▼ To create a public synonym

Required privileges: SQLEXECUTE and DBA

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a CREATE PUBLIC SYNONYM statement for the table, view, or synonym you want to represent with a synonym. For example, to create a public synonym called EMP for the table JACK.NEWEMPLOY, type:

```
CREATE PUBLIC SYNONYM EMP
FOR JACK.NEWEMPLOY
```

Renaming a synonym

You can rename a synonym by first dropping it with the DROP SYNONYM statement and then creating it with a different name. There is no SQL statement that automatically renames a synonym.

▼ To rename a synonym

Required privileges: SQLEXECUTE and DBA (for public synonyms) or synonym owner

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a DROP SYNONYM statement for the synonym you want to remove, and then submit a CREATE SYNONYM statement that specifies a new name for the synonym you just removed. Some examples follow.

– To rename a private synonym for the NEWEMPLOY table from EMP to NEW, type:

```
DROP SYNONYM EMP
CREATE SYNONYM NEW
FOR NEWEMPLOY
```

– To rename a public synonym for the NEWEMPLOY table from EMP to NEW, type:

```
DROP PUBLIC SYNONYM EMP
CREATE PUBLIC SYNONYM NEW
FOR NEWEMPLOY
```

Displaying information about synonyms

You can display information about synonyms by submitting a SELECT statement on the SYSSYNONYMS system table. Synonym information includes synonym name and the component it represents, owner and creator, and type (public or private).

▼ To display information about synonyms

Required privileges: SQLEXECUTE and DBA or SELECT on SYSSYNONYMS

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a SELECT statement for the SYSSYNONYMS system table. Some examples follow.

   – To display information about all synonyms in a database, type:

```
SELECT *
FROM SYSADM.SYSSYNONYMS
```

– To display information about all synonyms created by a specific account, for instance, by SYSADM, type:

```
SELECT *
FROM SYSADM.SYSSYNONYMS
WHERE SOWNER='SYSADM'
```

– To display just the synonyms and the tables or views they represent, type:

```
SELECT SNAME, STBL
FROM SYSADM.SYSSYNONYMS
```

– To display all synonyms for a specific user table (such as NEWEMPLOY), type:

```
SELECT *
FROM SYSADM.SYSSYNONYMS
WHERE STBL='NEWEMPLOY'
```

– To display all of the public synonyms in a database, type:

```
SELECT SNAME
FROM SYSADM.SYSSYNONYMS
WHERE ISPUBLIC='1'
```

Listing synonym names

You can list the synonym names by submitting a SELECT statement on the SYSSYNONYMS system table.

▼ To list synonym names

Required privileges: SQLEXECUTE and DBA or SELECT on SYSSYNONYMS

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit the following SELECT statement on the SYSSYNONYMS system table:

   ```
   SELECT SNAME,SOWNER
   FROM SYSADM.SYSSYNONYMS
   ```

Dropping a synonym

You can drop a private or public synonym by submitting a DROP SYNONYM statement. When you drop a synonym, any views or other synonyms that reference the synonym remain, but they become invalid.

▼ To drop a private synonym

Required privileges: SQLEXECUTE and DBA or synonym owner

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a DROP SYNONYM statement for the synonym you want to remove. For example, to remove a synonym called NEW, type:

   ```
   DROP SYNONYM NEW
   ```

▼ To drop a public synonym

Required privileges: SQLEXECUTE and DBA

1. Follow your site's procedure for connecting to your StorHouse database.

2. Submit a DROP PUBLIC SYNONYM statement for the public synonym you want to remove. For example, to remove a public synonym called EMP, type:

   DROP PUBLIC SYNONYM EMP

Chapter 10

Metadata backup

This chapter answers these questions about backing up metadata for a StorHouse database:

- Which files are backed up?
- When should you run a backup?
- Who can run a backup?
- What should you do before a backup?
- What happens during a backup?
- What should you do after a backup?
- Where are metadata backup files stored?
- How are backup files named?
- How many backup files can you keep?

Additionally, this chapter explains how to invoke the metadata backup utility and to display a list of metadata backup files. It also contains the error messages generated during a metadata backup. For information about recovering metadata from backup files, contact your FileTek customer support representative. For more information about disaster recovery features, refer to the *FileTek Recovery Strategies* manual, publication number 900117.

About metadata backup

Metadata is information that StorHouse/RM uses to manage a database. Backing up metadata is necessary to restore a database in the event of a catastrophic failure. With the metadata backup utility, you can back up metadata for one

database or multiple databases at a time. Users can query a database during a metadata backup.

Which files are backed up?

The metadata backup utility backs up all files in a system tablespace, including the following database components:

- System tables (files with .TBL extensions)
- System table indexes (files with .IDX extensions)
- System log files (files with .LOG extensions)

Use StorHouse Command Language CREATE BACKUP or ARCHIVE commands to back up segment files. Refer to the "Protecting User and System Files" section in the StorHouse *System Administrator's Guide* for more information about backing up StorHouse user files.

When should you run a backup?

StorHouse SQL Data Definition Language (DDL) statements—ALTER, CREATE, DROP, GRANT, and REVOKE—update metadata. Consider running a metadata backup after committing any DDL statements. Also, FileTek recommends that you back up metadata after each load, or if you run a set of loads during a "load window," back up metadata after the set completes.

Do not run a backup while a load is in progress. If any loads are active (unconfirmed, still running, failed and not yet restarted or aborted), an error occurs and the metadata backup utility stops. You can, however, force the utility (see "-FORCE" on page 10-9) to ignore the check for active loads and continue anyway. Be aware, however, that if you run loads or submit DDL statements during a metadata backup, you may have difficulty determining which transactions completed and which were queued, especially if your load stream contains DROP and CREATE statements.

Note: You can run a metadata backup when a database is down. The metadata backup utility will wait, however, if another process has an exclusive (write) lock on a system table.

Who can run a backup?

A StorHouse account with one of the following command privileges can run the metadata backup utility:

- OPERATOR
- SERVICE
- SYSTEM

Note that the SYSADM account has ALLPRIVILEGE (which includes the above privileges). Refer to the StorHouse *Command Language Reference Manual* for more information about command privileges.

What should you do before a backup?

Before invoking a metadata backup, set the following StorHouse system parameters for your site or use the default values if those are acceptable. See Appendix C, “System parameters for StorHouse/RM,” for more information about displaying and defining system parameters.

System parameters for metadata backup

System parameter	Description	Default
SQL_BKUP_ACCOUNT	StorHouse account used to write metadata backup files	SYSTEM
SQL_BKUP_FSET	File set to contain metadata backup files	RMMDBKUP
SQL_BKUP_GROUP	File access group for metadata backup files	RMMDBKUP

System parameters for metadata backup (continued)

System parameter	Description	Default
SQL_BKUP_LIMIT	Maximum number of backup file versions to keep for each database	10
SQL_BKUP_VSET	Volume set to contain metadata backup files	RMMDBKUP

What happens during a backup?

The metadata backup utility copies all metadata files in a database directory on UNIX to a single temporary file and then to a StorHouse file. Specifically, the metadata backup utility:

- Verifies that the specified database(s) exist
- Checks for active or at-checkpoint data loader operations
- Locks the database (read locks on system tables)
- Copies all metadata files to a single temporary file
- Unlocks the database
- Copies the temporary file to StorHouse
- Deletes the temporary file

Note that any DDL statements submitted or active loads already running during a backup are queued until after the metadata backup utility unlocks the database. The utility displays status messages at your terminal as well as fatal error messages at the StorHouse console.

What should you do after a backup?

FileTek recommends that you checkpoint StorHouse directory files after a metadata backup. The checkpoint feature, initiated with the Command Language CHECKPOINT command, takes a snapshot of StorHouse system files and writes them to optical disk or magnetic tape. This feature increases the speed

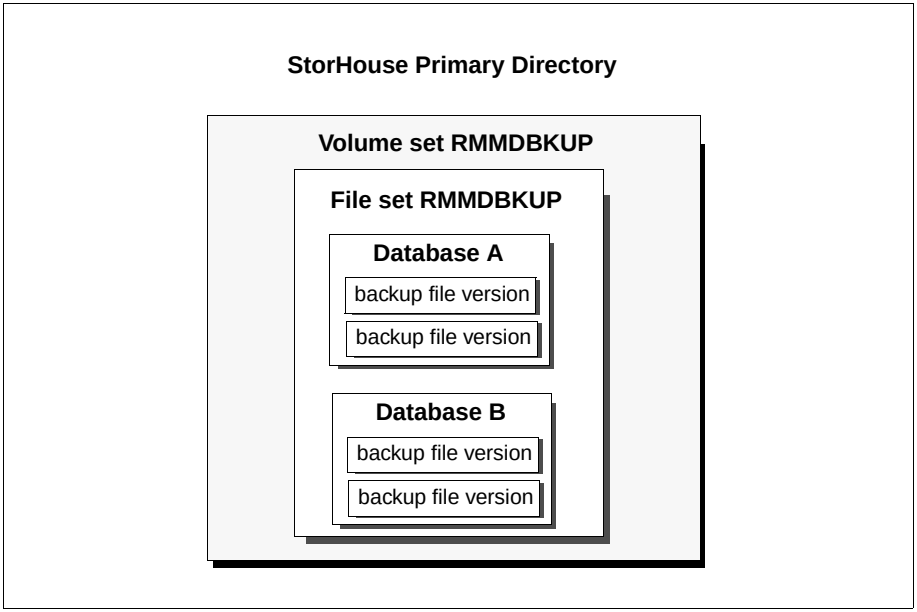
of recovery and ensures that StorHouse directory entries are in sync with the metadata backup files. This is especially important for loads and DROPs which affect both metadata and directory entries.

Also, consider making a backup or archive copy of each metadata backup file using the CREATE BACKUP or the ARCHIVE command. This enables you to export those copies from StorHouse and store them off-site for disaster recovery purposes. You should also back up or archive table data files, index files, and LOB subsegment files after a data load. Refer to the *StorHouse System Administrator's Guide* for more information about the StorHouse checkpoint, backup, and archive features.

Where are metadata backup files stored?

Metadata backup files are written to a file set in a volume set that is cataloged in the StorHouse primary directory. You specify the volume set and file set names using the SQL_BKUP_VSET and SQL_BKUP_FSET system parameters. The default name for both system parameters is RMMDBKUP. Each time you run a

backup, the metadata backup utility creates a backup file version for the specified database.



How are backup files named?

StorHouse files containing metadata have the following naming convention:

META_BKUP.dbname

where:

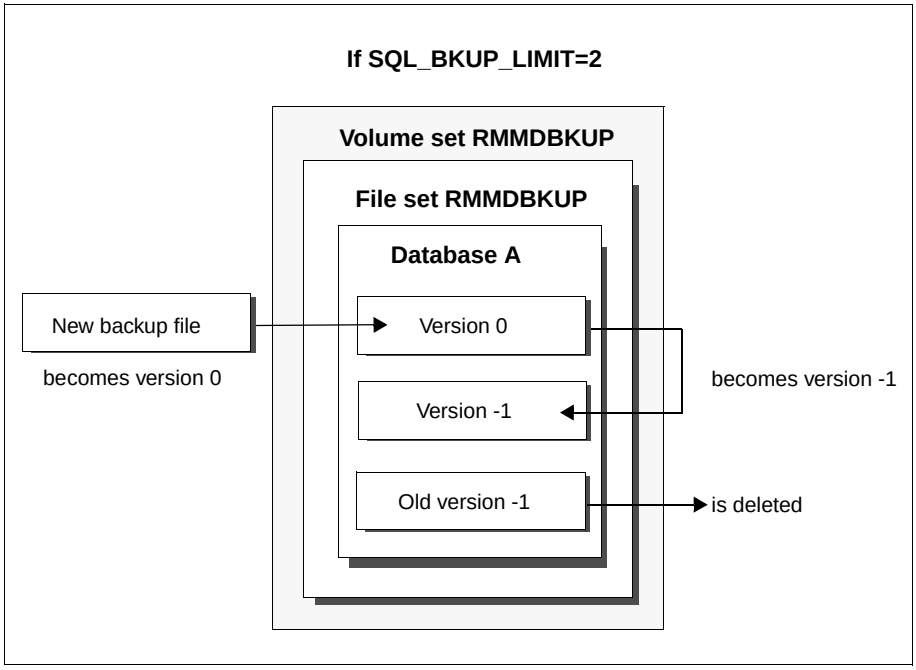
- META_BKUP is the file prefix
- dbname is the name of the database

For example, META_BKUP.CUSTOMER. StorHouse file version facilities maintain version numbers of metadata backup files.

How many backup files can you keep?

The SQL_BKUP_LIMIT system parameter specifies the maximum number of backup file versions to keep for each database. The maximum setting is 99 backup file versions for each database. The default setting is 10 backup file versions. In this StorHouse/RM release, only the latest file version (version 0) is used to recover metadata.

StorHouse automatically deletes old file versions to make room for new ones, and it renumbers the relative version numbers. For example, if SQL_BKUP_LIMIT=2 and the RMMDBKUP volume set and file set contain two backup file versions, then the next time you run a metadata backup, StorHouse deletes backup file version -1, renumbers backup file version 0 to version -1, and names the new backup file version 0.



Refer to the *StorHouse Concepts and Facilities Manual* for more information about file versions.

Invoking a metadata backup

You invoke the metadata backup utility via StorHouse. You can:

- Run a backup as needed by using the Command Language RUN command. See “Running a metadata backup as needed” on page 10-9 for more information about the RUN command.

- Schedule backups at specific frequencies and times by using the Command Language SCHEDULE command. See “Scheduling a metadata backup” on page 10-10 for examples.

You can also run the metadata backup utility with automated shell scripts and programs and with the StorHouse/Admin module of Control Center. Contact your FileTek customer support representative to run backup scripts and programs from UNIX. Refer to the *Control Center User Document Set* for more information about backing up as well as restoring metadata with StorHouse/Admin. If an error occurs during a backup, see page 10-11 for a list of error messages, explanations, and suggested user actions.

Running a metadata backup as needed

Use the StorHouse Command Language RUN command to invoke a metadata backup as needed. You can back up one database or multiple databases at a time. The format of RUN is:

RUN META_BKUP [-FORCE] database_name [database_name ...]

Argument	Description
-FORCE	(optional) Option to force the metadata backup utility to run even when loads are active. You'll receive warning messages if the metadata backup utility detects active loads.
database_name	(required) Name of the StorHouse database(s) to back up. Enclose database names with lowercase characters in double quotes.

▼ To run a metadata backup as needed using the RUN command

Required privileges: OPERATOR, SERVICE, or SYSTEM

1. If necessary, sign on to StorHouse.

2. At the command prompt (?), submit the RUN command and press J . Some examples follow.

   – To back up a database called Customer, type:

     RUN META_BKUP "Customer"

     In this example, the metadata backup utility stops if any loads are active (there's no -FORCE argument on the command statement). The database name Customer contains lowercase characters and is enclosed in double quotes.

– To back up two databases—SALES and INVENTORY—and to force the metadata backup utility to bypass the load check, type:

```
RUN META_BKUP -FORCE SALES INVENTORY
```

In this example, you receive warning messages if the metadata backup utility detects active loads, but the backup continues. Notice there's no comma separating the database names SALES and INVENTORY.

Scheduling a metadata backup

Use the StorHouse Command Language SCHEDULE command to schedule a metadata backup at specific frequencies and times. Refer to the StorHouse *Command Language Reference Manual* for complete information about the SCHEDULE command. Remember that the RUN command must be the last item in the command statement

▼ To schedule a metadata backup using the SCHEDULE command

Required privileges: SCHEDULE and OPERATOR or SERVICE or SYSTEM

1. If necessary, sign on to StorHouse.

2. At the command prompt (?), submit the SCHEDULE command and press J . Some examples follow.

   – To schedule an hourly metadata backup of the Customer database, executed by account SYSTEM and password STEM, beginning at 8:00 a.m. on January 1, 2000, type:

```
SCHEDULE /START=01-JAN-2000:08:00:00 /SCHEDULE=HOURLY
ACCOUNT=SYSTEM /PASSWORD=STEM !RUN META_BKUP
"Customer"
```

– To schedule a daily metadata backup of the Customer database, executed by account SYSTEM and password STEM, beginning at 6:00 p.m. on January 1, 2000, type:

```
SCHEDULE /START=01-JAN-2000:18:00:00 /SCHEDULE=DAILY
ACCOUNT=SYSTEM /PASSWORD=STEM !RUN META_BKUP
"Customer"
```

Metadata backup error messages

This section contains error messages that you may receive at your terminal during a metadata backup. Each message code begins with the prefix RMBR. In the message text, information between backward slashes \ \ represents variables.

RMBR001 StorHouse/RM software not installed

Explanation: You tried to run the metadata backup utility on a StorHouse system without StorHouse/RM software.

User Response: Contact your FileTek Account Representative.

RMBR002 No database name specified

Explanation: The RUN command didn't contain a database name. At least one database name is required.

User Response: On the RUN command, specify the name of the database you want to back up. Database names are case sensitive, so if the name contains lowercase characters, enclose it in double quotes.

RMBR003 Invalid command line option

Explanation: You must have typed run meta_bkup or an argument other than -FORCE.

User Response: Check the command line. Be sure the script name is in uppercase letters. Re-submit the RUN command with the appropriate arguments.

RMBR004 SM System Parameter not defined \parameter_name\

Explanation: A required StorHouse system parameter doesn't have a value. In order to run the metadata backup utility, the following system parameters require values: SQL_BKUP_ACCOUNT, SQL_BKUP_FSET, SQL_BKUP_GROUP, SQL_BKUP_LIMIT, and SQL_BKUP_VSET.

User Response: Set a value for the identified parameter_name. Refer to Appendix C, "System parameters for StorHouse/RM," for more information about setting system parameters.

RMBR005 Environment variable not defined \variable_name\

Explanation: A required environment variable—STHDBS or STHROOT—doesn't have a value.

User Response: Contact FileTek Customer Support.

RMBR006 Database does not exist \database_name\

Explanation: The database name supplied on the command line refers to a database that doesn't exist in StorHouse.

User Response: Resubmit the RUN command with the correct database name. Be sure to type the exact database name. Database names are case sensitive. Enclose names with lowercase letters in double quotes.

RMBR007 -FORCE not specified and checkpointed load operations detected

Explanation: The metadata backup utility detected successful load operations that haven't been confirmed or unsuccessful load operations that haven't been restarted or aborted.

User Response: Confirm, restart, or abort all load operations and then resubmit the RUN command. Or, you can use the -FORCE option to ignore the active load operations. Be aware, however, that if you force the utility to continue, you may have difficulty determining which transactions in the load stream completed and which were queued.

RMBR008 -FORCE not specified and load operation in progress

Explanation: The metadata backup utility detected a load operation that is running.

User Response: Wait until the load completes successfully and has been confirmed, and then resubmit the RUN command. Or, you can use the -FORCE option to ignore the active load operation. Be aware, however, that if you force the utility to continue, you may lose load checkpoint information and won't be able to restart the load if needed, or you might need to check the user tables and indexes to verify the state of the load operation.

RMBR009 Lock Manager error \database_name sqlcode\

Explanation: StorHouse could not lock the identified database.

User Response: Call FileTek Customer Support. Be sure to give the representative the sqlcode provided in the message text.

RMBR010 Error from 'tar' utility \errno\

Explanation: The UNIX utility that copies the metadata files and range index files from UNIX to a temporary file couldn't create the temporary file. The errno is a UNIX error number.

User Response: Contact FileTek Customer Support.

RMBR011 Error opening 'tar' file \filename errno\

Explanation: The metadata backup utility couldn't open the identified temporary file. The errno is a UNIX error number.

User Response: Contact FileTek Customer Support.

RMBR012 Error returned from SM operation \operation return_code\

Explanation: StorHouse either couldn't open or write to the metadata backup file, or the StorHouse account specified on the SQL_BKUP_ACCOUNT system parameter is invalid or doesn't have the correct privilege. The return_code is the StorHouse return code.

User Response: Refer to the StorHouse *Messages and Codes Manual* for more information about the return_code, then contact your StorHouse system administrator for help resolving the StorHouse account or file error.

RMBR013 Authentication failed for SQL_BKUP_ACCOUNT \account\

Explanation: The metadata backup utility couldn't obtain the password for the account specified on the SQL_BKUP_ACCOUNT system parameter or the account is invalid.

User Response: Contact your StorHouse system administrator.

RMBR016 SM Message \message_text\

Explanation: An error has been reported from a StorHouse API operation. The message_text contains the message returned by StorHouse. The RMBR016 error message is usually accompanied by other messages reporting the failing operation type and return code, for example RMBR012.

User Response: Refer to the User Response for the accompanying message(s).

RMBR019 Error changing directory \directory errno\

Explanation: A UNIX "change directory" (chdir) system call returned the error errno. The target directory is the StorHouse/RM database directory.

User Response: Determine the cause of the error from the UNIX error number.

RMBR020 Error checking loader status \sqlcode\

Explanation: A failure occurred while scanning the FileTek data loader checkpoint file. The error returned from the scanning routines is given by sqlcode.

User Response: Contact your FileTek customer support representative.

RMBR021 Error getting temporary directory \directory\

Explanation: The file etc/rdbtemp.data in the StorHouse/RM root directory could not be processed. This file is used to locate the directory to be used for the temporary file built during metadata backup processing.

User Response: Contact your FileTek customer support representative.

RMBR022 Backup of database \database\ Completed with failure

Explanation: The metadata backup operation did not complete successfully. The metadata for the indicated database does not have a usable backup.

User Response: Correct the problems indicated by other error messages and re-run the backup.

Listing metadata backup files

Use the StorHouse Command Language SHOW FILE command to list files in the metadata backup volume set and file set.

▼ To list all metadata backup files

Required privilege: SHOW

1. If necessary, sign on to StorHouse.

2. At the command prompt (?), submit the SHOW FILE command with the appropriate metadata backup volume set and file set names and press J . For example, to list all backup file versions in the RMMDBKUP volume set and file set, type:

   ```
   SHOW FILE * /VSET=RMMDBKUP /FSET=RMMDBKUP
   ```

Appendix

A

StorHouse system tables

This appendix contains reference information about StorHouse system tables, including:

- Where the system tables are located
- Who owns the system tables
- The purpose of each system table
- How to identify system tables
- Privileges for working with system tables
- Names and data types of the columns in each system table

About system tables

Each StorHouse database has its own collection of *system tables*. StorHouse/RM automatically creates these system tables for each new database. System tables contain information about a database. StorHouse/RM uses the system tables to record, verify, and conduct work. For example, StorHouse/RM updates various system tables when you create database components, and it reads system tables to verify that database components exist and that accounts are authorized to access them.

System table names

System table names, which you use to identify system tables in SQL statements, have the following format:

SYSADM.SYSname

where SYSADM is the owner of the system tables, SYS is a prefix for all system tables, and name is a descriptive title. For example, SYSADM.SYSTABLES is the name of the system table that describes all of the user tables, system tables, and views in a database.

Note: This publication refers to system tables in the SYS*name* format, assuming the SYSADM owner name.

System table names are *not* case sensitive, so you can use uppercase or lowercase letters when using system table names in SQL statements. For example, to select all information from SYSTABLES, you could type the SELECT statement in either of the following ways:

```
SELECT *
FROM SYSADM.SYSTABLES
```

or

```
SELECT *
FROM sysadm.systables
```

If you use the SYSADM account, you do not have to include the owner name as part of the system table name. For example:

```
SELECT *
FROM SYSTABLES
```

System table privileges

The SYSADM account has ALL privilege for *all* system tables in *all* databases. SYSADM can use the following SQL statements for all system tables: SELECT, INSERT, UPDATE, and DELETE.

An account with DBA privilege has ALL privilege for *all* system tables in *a* specific database. This account can use the following SQL statements for system tables in that database: SELECT, INSERT, UPDATE, and DELETE.

PUBLIC has SELECT privilege for *all* system tables—except SYSRANGES—in *all* databases. StorHouse accounts with SQLEXECUTE privilege can query all system tables except SYSRANGES. You must use SYSADM or have DBA or SELECT privilege to access the SYSRANGES system tables.

Caution: System tables contain critical database information. With the exception of the SYSSMUSERS system table, users should *never* use any SQL statement other than SELECT on the system tables. If you do not want PUBLIC access to system tables, revoke the SELECT privilege from PUBLIC in all databases. Be aware, however, that if you update or revoke the PUBLIC privilege for a system table, the metadata conversion process ignores the change. You must re-do the change in the converted database.

List of system tables

The following table lists and briefly describes the StorHouse system tables.

System table	Contains	See page
SYSCOLAUTH	Column update privileges granted to accounts	A-6
SYSCOLUMNS	One row for each column of every table in a database	A-7
SYSDBAUTH	Database privileges for accounts	A-8
SYSINDEXES	One row for each indexed column	A-10
SYSPACKAGE	One row for each package in a database	A-12
SYSPACKSTMT	One or more rows for each statement in a package	A-14
SYSRANGES	Range indexes	A-16
SYSSMUSERS	One row for each account with a default tablespace, plus a default tablespace entry for PUBLIC	A-19
SYSSTAT_COL	Statistics for each indexed column of a segment	A-20
SYSSTAT_HIST	Spread and frequency of each histogram bucket	A-21
SYSSTAT_IDX	Average number of rows returned for queries that use value indexes	A-21
SYSSTAT_SMATRIX	Spread encoding matrixes used to calculate the spread of non-numeric values	A-23
SYSSTHFILES	StorHouse file and group names for segment files	A-24
SYSSTHSEGMENTS	Information about segments	A-25
SYSSTHSPACES	Storage parameters for user tablespaces	A-26
SYSSYNONYMS	One row for each synonym in a database	A-29

System table	Contains	See page
SYSTABAUTH	Table privileges held by accounts	A-30
SYSTABLES	One row for each user table, system table, and view in a database	A-32
SYSTBLSPACES	One row for each tablespace in a database	A-34
SYSVIEWS	One row for each view in a database	A-35

The following system tables are for future use and not described in this appendix:

- SYSDATATYPES
- SYSDBLINKS
- SYSIDXSTAT
- SYSTBLSTAT
- SYS_CHKCOL_USAGE
- SYS_CHK_CONSTRS
- SYS_KEYCOL_USAGE
- SYS_REF_CONSTRS
- SYS_TBL_CONSTRS

SYSCOLAUTH

The SYSCOLAUTH system table indicates the individual system table or system table view columns that accounts can update. StorHouse/RM updates this system table when you grant and revoke the UPDATE privilege for system table columns.

Column name	Data type	Description
GRANTOR	VARCHAR(32) NOT NULL	Account ID of the account who granted the column privileges.
GRANTEE	VARCHAR(32) NOT NULL	Account ID of the account who holds the column privileges.
TBLOWNER	VARCHAR(32) NOT NULL	Account ID of the owner of the table or view on which the column privileges were granted.
TBL	VARCHAR(32) NOT NULL	Name of the table or view on which the column privileges were granted.
COL	VARCHAR(32) NOT NULL	Name of the column to which the UPDATE privilege applies.
UPD	VARCHAR(1) NOT NULL	Grantee's UPDATE privilege for the column in the table or view. Values are: y Can update this column in the system table [ ] (Blank) Indicates no UPDATE privilege
REF	VARCHAR(1) NOT NULL	For future use.

SYSCOLUMNS

The SYSCOLUMNS system table contains one row for each column in every table in a database. When you create a user table, StorHouse/RM inserts a row into this system table for each column in the table.

Column name	Data type	Description
ID	INTEGER NOT NULL	ID assigned to the column.
COL	VARCHAR(32) NOT NULL	Name of the column.
TBL	VARCHAR(32) NOT NULL	Name of the table that contains the column.
OWNER	VARCHAR(32) NOT NULL	Account ID of the owner of the table.
COLTYPE	VARCHAR(10) NOT NULL	Data type of the column. Values are: BINARY, BLOB, CHARACTER, CLOB, DATE, DOUBLE, FLOAT, INTEGER, NUMERIC, REAL, SMALLINT, TIME, TIMESTAMP, VARBINARY, VARCHAR
WIDTH	INTEGER NOT NULL	Length of the column, or in the case of data type NUMERIC or DECIMAL, the precision of the data type.
SCALE	INTEGER NULL	Scale of data type NUMERIC or DECIMAL.
NULLFLAG	VARCHAR(1) NOT NULL	Null indicator for the column. Values are: Y Can be null N Cannot be null

Column name	Data type	Description
DFLT_VALUE	VARCHAR(250) NULL	Default value for the column, if defined in a CREATE TABLE statement.
LOB_TSID	INTEGER NOT NULL	ID of the user tablespace to which the LOB column is assigned. If the column is a non-LOB column, then LOB_TSID has a value of -1. If the LOB column is assigned to the same tablespace as the table, then LOB_TSID has the ID of the user tablespace for the table.
LOB_INLINE_MAX	INTEGER NULL	Maximum length of an in-line LOB for this column. A NULL indicates the column is a non-LOB data type or the LOB is always stored out-of-line. A 0 means the length was omitted on the INLINE clause; therefore, the default maximum in-line length is 32705.
LOB_STORE_WITH	INTEGER NULL	ID of the LOB column to share storage with this LOB column. The first column in a table has column ID 0. A NULL indicates one of the following: ■ Column is a non-LOB data type ■ LOB column does not share storage with another LOB column ■ LOB is always stored in-line

SYSDBAUTH

The SYSDBAUTH system table contains database privileges—DBA, RESOURCE, and SCAN—for accounts. StorHouse/RM maintains this system table when you grant and revoke database privileges.

Column name	Data type	Description
GRANTEE	VARCHAR(32) NOT NULL	Account ID of the account who holds the database privileges.
DBA_ACC	VARCHAR(1) NOT NULL	Grantee's DBA privilege for the database. Values are: y Can create and select database components [ ] (Blank) Cannot create and select database components
RES_ACC	VARCHAR(1) NOT NULL	Grantee's RESOURCE privilege for the database. Values are: y Can create database components [ ] (Blank) Cannot create database components
SCN_ACC	VARCHAR(1) NOT NULL	Grantee's SCAN privilege for the database. Values are: y Can scan user tables [ ] (Blank) Cannot scan user tables

SYSINDEXES

The SYSINDEXES system table contains one row for each indexed column of every system table and user table in a database. StorHouse/RM inserts a row (or multiple rows in the case of a compound index) into this system table when you create an index and deletes a row(s) when you drop an index.

For a compound index, StorHouse/RM inserts one row in this table for every column in the compound index. If the compound index is composed of two columns, then the SYSINDEXES system table contains two rows for that index.

Column name	Data type	Description
ID	INTEGER NOT NULL	ID assigned to the index.
TSID	INTEGER NOT NULL	ID of the tablespace to which the index is assigned. For a system table index, the value 0 indicates the system tablespace ID.
IDXNAME	VARCHAR(32) NOT NULL	Name of the index.
IDXOWNER	VARCHAR(32) NOT NULL	Name of the owner of the index.
TBL	VARCHAR(32) NOT NULL	Name of the table on which the index is defined.
TBLOWNER	VARCHAR(32) NOT NULL	Name of the owner of the table on which the index is defined.
COLNAME	VARCHAR(32) NOT NULL	Name of the column on which the index is defined.
IDXTYPE	VARCHAR(1) NOT NULL	Unique indicator. Values are: U Unique index (no duplicate column values) D Non-unique (duplicate) index

Column name	Data type	Description
IDXORDER	VARCHAR(1) NOT NULL	Order of values in the index. Values are: A Ascending values D Descending values
IDXCOMPRESS	VARCHAR(1) NOT NULL	Status of the index. Values are: N Normal (complete) C Deferred (incomplete)
IDXMETHOD	VARCHAR(1) NOT NULL	Type of index. Values are: B B-tree for system table index or value index for user table index H Hash index for user tables R Range index for user tables
IDXSEQ	INTEGER NOT NULL	Column sequence number within the index. Sequence numbers start with 0.
COMPRESSION	SMALLINT NOT NULL	For future use.

SYSPACKAGE

The SYSPACKAGE system table contains one row for each package in a database. The combination of RDBCOLID, PKGID, and VERSION_NAME uniquely identifies a StorHouse package.

Column name	Data type	Description
RDBNAM	CHAR(18) NOT NULL	Name of the StorHouse database.
RDBCOLID	CHAR(18) NOT NULL	Identifier of the package collection.
PKGID	CHAR(18) NOT NULL	Identifier of the specific StorHouse package.
PKGCNSTKN	CHAR(16) NOT NULL	Consistency token, which verifies synchronization of the requestor application and the database package. Packages with the same fully qualified package name cannot have the same consistency token.
CCSIDSBC	SMALLINT NULL	Coded character set identifier for single-byte characters. This column is not used in this release.
CCSIDDBC	SMALLINT NULL	Coded character set identifier for double-byte characters. This column is not used in this release.
CCSIDMBC	SMALLINT NULL	Coded character set identifier for mixed-byte characters. This column is not used in this release.
VERSION_NAME	VARCHAR(254) NULL	Version identifier associated with the package.
REPLACED_PKG_ NAME	VARCHAR(254) NULL	Version identifier of the package that was replaced by the current package.
DECIMAL_ PRECISION	SMALLINT NULL	Default precision used during decimal arithmetic processing.

Column name	Data type	Description
DEFAULT_ RDBCOLID	CHAR(18) NULL	Default collection identifier used (when necessary) to complete component names in SQL statements that are bound into this package.
AUTH_OPT	SMALLINT NULL	Package authorization option specifying whether the existing package authorizations are maintained or revoked when a package is replaced.
DEFAULT_CHAR_ SUBTYPE	SMALLINT NULL	Default SQL character subtype used if a character column is defined by an SQL CREATE TABLE or ALTER TABLE statement with no explicit subtype specification. Note that ALTER TABLE is not supported in this release.
ISOLATION_LEVEL	SMALLINT NULL	Isolation level used when SQL statements in the package are executed. This isolation level may be overridden by the target relational database run-time mechanism.
OWNER_ID	CHAR(8) NULL	Account ID of the package owner.
BLOCK_PROTOCOL	SMALLINT NULL	Controller of the query block protocol used when a query is opened. When specified in the BGNBND command, it controls the protocols used by all queries in the package unless overridden by the OPNQRY command.
RELEASE_OPTION	SMALLINT NULL	Option for releasing the package execution resources and the associated serialization or sharing of locks.
DATE_FORMAT	SMALLINT NULL	Default date format used in SQL statements.
DECIMAL_ DELIMITER	SMALLINT NULL	Default character string used as the decimal delimiter in SQL statements.

Column name	Data type	Description
STRING_DELIMITER	SMALLINT NULL	Characters that delimit strings and delimited SQL identifiers in SQL statements.
TIME_FORMAT	SMALLINT NULL	Default time format used in SQL statements.
TITLE	VARCHAR(255) NULL	Brief description of the package.

SYSPACKSTMT

The SYSPACKSTMT system table contains one or more rows for each statement in a package.

Column name	Data type	Description
RDBNAM	CHAR(18) NOT NULL	Name of the StorHouse database.
RDBCOLID	CHAR(18) NOT NULL	Identifier of the package collection.
PKGID	CHAR(18) NOT NULL	Identifier of the StorHouse package.
PKGCNSTKN	CHAR(16) NOT NULL	Consistency token, which verifies synchronization of the requestor application and the database package. Packages with the same fully qualified package name cannot have the same consistency token.
PKGSN	SMALLINT NOT NULL	StorHouse package section number.

Column name	Data type	Description
STMTSEQNO	SMALLINT NOT NULL	Statement sequence number. If the STATEMENT column contains text for the SQL statement that the row represents, then STMTSEQNO specifies the sequence number of that text within the entire SQL statement.
STATEMENT	VARCHAR(254) NOT NULL	All or a portion of the text for the SQL statement that the row represents.
STATEMENT_TYPE	SMALLINT NULL	Type of statement. Values are: 1 - DECLARE 2 - SELECT 3 - INSERT 4 - UPDATE 5 - DELETE 6 - CREATE 7 - GRANT 8 - REVOKE 9 - DYNAMIC 10 - COMMIT 11 - ROLLBACK
TYPDEFNAM	CHAR(9) NULL	Name of a data type definition.
CCSIDSBC	SMALLINT NULL	Coded character set identifier for single-byte characters. This column is not used in this release.
CCSIDDBC	SMALLINT NULL	Coded character set identifier for double-byte characters. This column is not used in this release.
CCSIDMBC	SMALLINT NULL	Coded character set identifier for mixed-byte characters. This column is not used in this release.

SYSRANGES

The SYSRANGES system tables contain range indexes for each data type (except BLOB and CLOB). PUBLIC does not have SELECT privilege on these system tables. Only SYSADM or an account with DBA privilege or SELECT privilege can access these system tables. The convention for naming the range index system tables is:

SYSRANGES_datatype

where datatype is the name of the column data type. The following table maps column data types to range index system tables.

Data type	Length	System table name
BINARY	1-32	SYSRANGES_VARBINARY_SHORT
BINARY	33-256	SYSRANGES_VARBINARY_LONG
CHAR	1-32	SYSRANGES_VARCHAR_SHORT
CHAR	33-256	SYSRANGES_VARCHAR_LONG
DATE		SYSRANGES_DATE
DOUBLE PRECISION		SYSRANGES_DOUBLE
INTEGER		SYSRANGES_INTEGER
NUMERIC		SYSRANGES_DECIMAL
REAL		SYSRANGES_REAL
SMALLINT		SYSRANGES_SMALLINT
TIME		SYSRANGES_TIME
TIMESTAMP		SYSRANGES_TIMESTAMP
VARBINARY	1-32	SYSRANGES_VARBINARY_SHORT
VARBINARY	33-256	SYSRANGES_VARBINARY_LONG

Data type	Length	System table name
VARCHAR	1-32	SYSRANGES_VARCHAR_SHORT
VARCHAR	33-256	SYSRANGES_VARCHAR_LONG

The SYSRANGES system tables contain the following columns.

Column name	Data type	Description
IDXID	INTEGER NOT NULL	ID assigned to the index.
COLSEQ	INTEGER NOT NULL	Sequence number of the column in the index. Values are 0 to 15.
SEGID	INTEGER NOT NULL	ID of the segment.
NULLID	CHAR(1) NOT NULL	Flag indicating whether any value in the column is NULL. Values are: Y At least one value in the column is NULL. N None of the values in the column is NULL.
LOWVAL	Determined by the data type of the range index system table	Lowest data value in the segment. A NULL value is not considered as a LOWVAL unless all values in the column are NULL.
HIVAL	Determined by the data type of the range index system table	Highest data value in the segment. A NULL value is not considered as a HIVAL unless all values in the column are NULL.

The following table contains the data type of the LOWVAL and HIVAL columns for each SYSRANGES system table.

SYSRANGES system tables	LOWVAL and HIVAL data types
SYSRANGES_VARCHAR_SHORT	VARCHAR(32)
SYSRANGES_VARCHAR_LONG	VARCHAR(256)
SYSRANGES_VARBINARY_SHORT	VARBINARY(32)
SYSRANGES_VARBINARY_LONG	VARBINARY(256)
SYSRANGES_SMALLINT	SMALLINT
SYSRANGES_INTEGER	INTEGER
SYSRANGES_DECIMAL	DECIMAL(31,0)*
SYSRANGES_REAL	REAL
SYSRANGES_DOUBLE	DOUBLE
SYSRANGES_DATE	DATE
SYSRANGES_TIME	TIME
SYSRANGES_TIMESTAMP	TIMESTAMP

* The actual scale of the stored value is determined from the column definition in the SYSCOLUMNS system table.

SYSSMUSERS

The SYSSMUSERS system table contains an entry for each account with a default user tablespace. It also contains an entry for PUBLIC, which designates the default user tablespace for the database. Accounts that are excluded from this table have no account default user tablespace. You can maintain this system table.

Caution: When inserting, updating, or deleting a row in the SYSSMUSERS system table, remember to type the account ID in uppercase. Also, if you delimited the tablespace name when you created the user tablespace, and the name contains lowercase letters (for instance, CREATE TABLE SPACE "Mar2000"), be sure to type the name in the same case (Mar2000, not MAR2000 or mar2000). If you didn't delimit the tablespace name, type it in uppercase letters.

Column name	Data type	Description
ACCOUNTID	CHAR(12) NOT NULL	Account ID of the StorHouse account or PUBLIC for the database default tablespace.
DEFAULT_TS	CHAR(32) NOT NULL	Name of the default user tablespace. StorHouse uses this tablespace when the account creates a user table and does not specify a tablespace name.

SYSSTAT_COL

The SYSSTAT_COL system table contains statistics for each indexed column of a segment.

Column name	Data type	Description
TBLID	INTEGER NOT NULL	ID of the user table.
COLID	INTEGER NOT NULL	ID of the indexed column.
SEGID	INTEGER NOT NULL	ID of the segment.
LOW_VAL	VARCHAR(40)* NULL	Lowest value in the column sample.
LOW_VAL2	VARCHAR(40)* NULL	Second lowest value in the column sample.
HI_VAL2	VARCHAR(40)* NULL	Second highest value in the column sample.
HI_VAL	VARCHAR(40)* NULL	Highest value in the column sample.
HISTID	INTEGER NOT NULL	ID of the histogram.
NUM_BUCKETS	SMALLINT NOT NULL	Total number of buckets in the histogram.
NULLVAL_SEL	REAL NOT NULL	Selectivity of NULL values in the column sample.
SMATID	INTEGER NULL	ID of the spread matrix used to calculate the spread for non-numeric values.
IO_FACTOR	REAL NOT NULL	Expected number of page I/O operations required to retrieve one row.

* The data stored in these VARCHAR columns may be of any data type. StorHouse/RM performs any necessary conversions and stores the values in literal form. BINARY or VARBINARY, however,

is stored without conversion and may contain unprintable characters. CHAR, VARCHAR, BINARY, and VARBINARY values longer than 40 bytes are truncated.

SYSSTAT_HIST

The SYSSTAT_HIST system table contains the spread and frequency of each histogram bucket.

Column name	Data type	Description
HISTID	INTEGER NOT NULL	ID of the histogram.
BUCKET_SEQ	SMALLINT NOT NULL	Sequence number of the histogram bucket.
NUM_DISTINCTVAL	SMALLINT NOT NULL	Number of distinct column values in the bucket, estimated from a reservoir sample.
LOW_VAL	DOUBLE PRECISION NOT NULL	Lowest value in the histogram bucket.
SPREAD	DOUBLE PRECISION NOT NULL	Spread (difference) between the highest and lowest values in the histogram bucket.
FREQUENCY	SMALLINT NOT NULL	Total frequency (number of records) in the histogram bucket.

SYSSTAT_IDX

The SYSSTAT_IDX system table contains the average number of rows returned for queries that use value indexes, that is, queries that contain an equals predicate for one or more columns.

Column name	Data type	Description
IDXID	INTEGER NOT NULL	ID of the value index.
SEGID	INTEGER NOT NULL	ID of the segment.
IN_SORT	CHAR(1) NULL	Flag indicating whether the table rows are in sorted order of the indexed column's key values. Valid values: Y In sorted order N Not in sorted order
COL1_CARD	DOUBLE PRECISION NOT NULL	Average number of result rows for a query with an equals predicate on column 1.
COL12_CARD	DOUBLE PRECISION NULL	Average number of result rows for a query with an equals predicate on columns 1 and 2.
COL123_CARD	DOUBLE PRECISION NULL	Average number of result rows for a query with an equals predicate on columns 1, 2, and 3.
COL1234_CARD	DOUBLE PRECISION NULL	Average number of result rows for a query with an equals predicate on columns 1, 2, 3, and 4.

SYSSTAT_SMATRIX

The SYSSTAT_SMATRIX system table stores spread encoding matrixes used to calculate the spread of non-numeric values.

Column name	Data type	Description
SMATID	INTEGER NOT NULL	ID of the spread encoding matrix.
SMAT_BASE	SMALLINT NOT NULL	Number of marker array entries that are set to 1.
SMAT_MATRIX	BINARY(32) NOT NULL	Spread encoding matrix.

SYSSTHFILES

The SYSSTHFILES system table contains the StorHouse file name and group name for each table data file, index file, and LOB subsegment file. A StorHouse engine updates this system table after a confirmed load, after an index load and a merge operation, and when you drop user tables and indexes. A StorHouse engine uses this system table to locate the data during query processing.

Column name	Data type	Description
ID	INTEGER NOT NULL	ID of the table or index.
OBJECT_TYPE	CHAR(1) NOT NULL	Type of database component. Values are: T Table data H Hash index V Value index L LOB data
SUBSEGID	INTEGER	ID of the LOB subsegment.
SEGID	INTEGER NOT NULL	ID of the segment.
FILENAME	VARCHAR(56) NOT NULL	Name of the StorHouse file.
GROUPID	VARCHAR(8) NOT NULL	Name of the StorHouse file access group.

SYSSTHSEGMENTS

The SYSSTHSEGMENTS system table contains information about segments. A StorHouse engine updates this system table after a data load, replace, or merge operation and when you drop tables.

Column name	Data type	Description
TBLID	INTEGER NOT NULL	ID of the user table.
SEGID	INTEGER NOT NULL	ID of the segment.
ROW_COUNT	DOUBLE PRECISION	Number of rows in the table data file. A NULL value indicates that the row count was not collected when the segment was loaded.
ROWS_PER_PAGE	REAL	Average row count per page in the table data file. A NULL value indicates that the average row count was not collected when the segment was loaded.
LOAD_CONF_TIME	TIMESTAMP	Date and time that the operation was confirmed. A NULL value indicates that the load confirmation time was not recorded when the segment was loaded.
INVALID_FLAG	CHAR(1) NOT NULL	Flag indicating that the segment has been invalidated. Values are: Y Segment is invalidated. N Segment is loaded and accessible.

Column name	Data type	Description
INVALID_ TIME	TIMESTAMP	Date and time that the segment was invalidated. A NULL value indicates that the segment has never been invalidated.
SEGMENT_TAG	VARCHAR(40) NOT NULL	Name of the segment (either the segment tag provided on the SEGMENT clause of a LOAD statement or the load ID if the SEGMENT clause was omitted). For data loaded before StorHouse/RM release 2.2, there is no SEGMENT_TAG, that is, the value is 0 length.

SYSSTHSPACES

The SYSSTHSPACES system table contains one row for each subspace in a user tablespace. This system table identifies the storage specifications for each user tablespace. StorHouse/RM maintains this system table when you create, alter, and drop user tablespaces.

Column name	Data type	Description
TSID	INTEGER NOT NULL	ID of the user tablespace.
TSSUBSPACE_NUM	INTEGER NOT NULL	Number assigned to the subspace.
TSOBJECT_TYPE	CHAR(1) NOT NULL	Type of component allowed in this subspace. Values are: blank — (default) All components (no restriction) T — Table data only H — Hash index only V — Value index only L — LOB data only

Column name	Data type	Description
TSVSET	CHAR(8) NOT NULL	Name of the volume set that contains components in this subspace.
TSFSET	CHAR(8) NOT NULL	Name of the file set that contains components in this subspace.
TSVTF	CHAR(8) NOT NULL	Vulnerability time factor (VTF) for components in this subspace. Values are: DEFAULT (default) Use the value of the StorHouse VTF system parameter. NOW Write the file to the performance buffer first and then copy it immediately to its file set. NEXT Write the file to the performance buffer and copy it to its file set during the next StorHouse write-back operation. DIRECT Bypass the performance buffer and write the file directly to the file set. (DF and map extents, however, are always written to the performance buffer as well as to the file set.)
TSATF	SMALLINT NOT NULL	Access time factor (ATF) for components in this subspace. Values are: 0 (default) Use the value of the StorHouse ATF system parameter. 1 Short access time is important. 2 Short access time is moderately important. 3 Short access time is minimally important.

Column name	Data type	Description
TSEDC	CHAR(1) NOT NULL	Error Detection Code option for components in this subspace. Values are: D (default) Use the value of the StorHouse EDC system parameter. Y Use EDC. N Do not use EDC.
TSGROUP	CHAR(8) NOT NULL	Name of the StorHouse file access group for components in this subspace. The default file access group is STH.
TSMAX_EXT_SIZE	SMALLINT NOT NULL	Maximum size (in megabytes) of extents in this subspace. The size range is 1 to the maximum surface size for the VSET. The default 0 uses 100 MB for LOB subsegment files or the value of the applicable StorHouse system parameter: ■ SQL_MAX_EXT_DATA for table data files ■ SQL_MAX_EXT_HASH for hash index files ■ SQL_MAX_EXT_VAL for value index files
TSHOLD	SMALLINT NOT NULL	Number of days to keep data extents in the performance buffer. The range of days is 0 to 32767. For LOB subsegment files, the default 0 uses 0 days. For table data files and index files, the default 0 uses the value of the applicable StorHouse system parameter: ■ SQL_HOLD_DATA for table data files ■ SQL_HOLD_INDX for index files
TSHOLD_SPECIAL	SMALLINT NOT NULL	Number of days to keep DF and map extents in the performance buffer. The range of days is 0 to 32767. The default 0 uses the value of the SQL_HOLD_SPECIAL system parameter.

SYSSYNONYMS

The SYSSYNONYMS system table contains one entry for each synonym in a database. StorHouse/RM maintains this system table when you create and drop synonyms.

Column name	Data type	Description
SNAME	VARCHAR(32) NOT NULL	Name of the synonym.
SOWNER	VARCHAR(32) NOT NULL	Account ID of the owner of the synonym.
SCREATOR	VARCHAR(32) NOT NULL	Account ID of the creator of the synonym.
STBL	VARCHAR(32) NOT NULL	Name of the table, view, or synonym for which the synonym was created.
STBLOWNER	VARCHAR(32) NOT NULL	Name of the owner of the table, view, or synonym for which the synonym was created.
ISPUBLIC	SMALLINT NOT NULL	Type of synonym. Values are: 0 Private synonym 1 Public synonym
SREMDB	VARCHAR(32) NULL	Remote database name.

SYSTABAUTH

The SYSTABAUTH system table contains the table and view privileges for accounts. StorHouse/RM maintains this system table when you grant and revoke database component privileges.

Column name	Data type	Description
GRANTOR	VARCHAR(32) NOT NULL	Account ID of the account that granted the table privileges.
GRANTEE	VARCHAR(32) NOT NULL	Account ID of the account that holds the table privileges. The grantee can also be PUBLIC.
TBLOWNER	VARCHAR(32) NOT NULL	Account ID of the owner of the table or view on which the table privileges were granted.
TBL	VARCHAR(32) NOT NULL	Name of the table or view on which the table privileges were granted.
INS	VARCHAR(1) NOT NULL	Grantee's INSERT privilege for loading data into the user table or inserting data into the system table or system table view. Values are: y Can insert rows into the system table or load data into the user table g Can insert or load rows, WITH GRANT OPTION [ ] (Blank) Cannot insert or load rows
DEL	VARCHAR(1) NOT NULL	Grantee's DELETE privilege for the system table or system table view. Values are: y Can delete rows from the system table g Can delete rows, WITH GRANT OPTION [ ] (Blank) Cannot delete rows from the system table

Column name	Data type	Description
UPD	VARCHAR(1) NOT NULL	Grantee's UPDATE privilege for the system table or system table view. Values are: y Can update rows in the system table g Can update rows, WITH GRANT OPTION [ ] (Blank) Cannot update rows in the system table * Privilege exists for some, but not all columns in the table
SEL	VARCHAR(1) NOT NULL	Grantee's SELECT privilege for the user table, user table view, system table, or system table view. Values are: y Can query the table or view g Can query the table or view, WITH GRANT OPTION [ ] (Blank) Cannot query the table or view
NDX	VARCHAR(1) NOT NULL	Grantee's INDEX privilege for the user table or system table. Values are: y Can create an index for the table g Can create an index, WITH GRANT OPTION [ ] (Blank) Cannot create an index for the table
ALT	VARCHAR(1) NOT NULL	For future use.
REF	VARCHAR(1) NOT NULL	For future use.

SYSTABLES

The SYSTABLES system table contains one row for each user table, system table, and view in a database. StorHouse/RM inserts rows for system tables when you create a database and inserts rows for user tables and views when you create them. StorHouse/RM deletes rows when you drop user tables or views. In the following table, the word *table* refers to system tables, user tables, and views.

Note: StorHouse/RM prevents updates to the rows for system tables.

Column name	Data type	Description
ID	INTEGER NOT NULL	ID assigned to the table.
TBL	VARCHAR(32) NOT NULL	Name of the table.
CREATOR	VARCHAR(32) NOT NULL	Account ID of the creator of the table.
OWNER	VARCHAR(32) NOT NULL	Account ID of the owner of the table.
TBLTYPE	VARCHAR(1) NOT NULL	Type of table. Values are: T Table V View
TBLPCTFREE	INTEGER NOT NULL	For future use.
TBLSPACEID	INTEGER NOT NULL	ID of the user tablespace to which the user table is assigned.
HAS_PCNSTRS	VARCHAR(1) NOT NULL	For future use.
HAS_FCNSTRS	VARCHAR(1) NOT NULL	For future use.
HAS_CCNSTRS	VARCHAR(1) NOT NULL	For future use.

Column name	Data type	Description
HAS_UCNSTRS	VARCHAR(1) NOT NULL	For future use.
TBL_STATUS	VARCHAR(1) NOT NULL	Not used.
RSSID	INTEGER NOT NULL	Identifier for the storage system. Values are: 1 StorHouse 0 System or temporary tablespace
SM_LOAD_ SEGMENT	INTEGER NOT NULL	Next segment ID to use for loading.
COMPRESSION	SMALLINT NOT NULL	For future use.

SYSTBLSPACES

The SYSTBLSPACES system table contains one row for each tablespace in a database. StorHouse/RM inserts a row into this system table when you create a user tablespace and deletes a row when you drop a user tablespace.

Column name	Data type	Description
ID	INTEGER NOT NULL	ID of the tablespace.
TSNAME	VARCHAR(32) NOT NULL	Name of tablespace.
STORAGE_MANAGER	VARCHAR(32) NOT NULL	Identifier for the storage system. Values are: STH StorHouse DICT System tablespace TEMP Temporary tablespace
NUM_SUBSPACES	INTEGER NOT NULL	Number of subspaces in the user tablespace.

SYSVIEWS

The SYSVIEWS system table contains one row for each view in a database. StorHouse/RM maintains this system table when you create and drop views.

Column name	Data type	Description
VIEWNAME	VARCHAR(32) NOT NULL	Name of the view.
CREATOR	VARCHAR(32) NOT NULL	Account ID of the creator of the view.
OWNER	VARCHAR(32) NOT NULL	Account ID of the owner of the view.
SEQ	INTEGER NOT NULL	Sequence number used for packing and unpacking view text.
VIEWTEXT	VARCHAR(900) NOT NULL	SQL text of the view.

Appendix B

System limits

The following table summarizes the limits for StorHouse databases.

Item	Limit
Length of a column name	32 characters
Length of a database name	32 characters (16 for DRDA)[1]
Length of a default value for a column	32 characters
Length of a file set name	8 characters
Length of an owner name	32 characters
Length of a table name	32 characters
Length of a StorHouse account ID	12 characters
Length of a StorHouse file name	56 characters
Length of a user tablespace name	32 characters
Length of a volume set name	8 characters
Length of an index name	32 characters
Number of columns in a compound index	16 columns
Number of columns in a table	500 columns
Number of databases	No limit
Number of active databases at a time	100 databases
Number of engines that can run concurrently	value of SQL_SESSIONS
Number of indexes in a database	No meaningful limit[2]

Item	Limit
Number of indexes in a table	150 indexes
Number of table files, index files, and LOB subsegment files in all user tables	7,000,000 files
Number of tablespaces	No meaningful limit[2]
Number of subspaces in a user tablespace	No meaningful limit[2]
Number of user and system tables in a database	none
Number of read locks on a user or system table	100 locks
Number of user and system tables that can be accessed in a database at a time	100 tables
Number of range indexes that can be referenced in a single query block	32 range indexes
Number of concurrently open cursors	32 cursors
Size of a row in a table	32,705 bytes
Size of an in-line LOB	32,705 bytes
Size of an out-of-line LOB	2**31-9
Size of a column in an index	256 bytes
Size of a key in a compound index	4,096 bytes
Size of a table data file	100 gigabytes
Size of a StorHouse file extent	Configurable (but cannot exceed media restriction)
Size of an SQL statement	10,000 bytes

[1] If a database name exceeds 16 characters, StorHouse/RM uses an alias in the StorHouse file name.

[2] Index, tablespace, and subspace numbers are limited only by internal structures with maximum values like 2**31 or 10**9.

The following table summarizes the limits for StorHouse database data types.

Item	Limit
BINARY	256 bytes
BLOB	2**31-9
CHAR	256 characters
CLOB	2**31-9
DATE	0001-01-01 to 9999-12-31
DOUBLE PRECISION	Sign bit, 11-bit exponent, 52-bit fraction
INTEGER	-2,147,483,648 (-2^{31}) to 2,147,483,647 (2^{31}-1)
NUMERIC	1-10^{31} to 10^{31} -1
REAL	Sign bit, 8-bit exponent, 23-bit fraction
SMALLINT	-32,768 to 32,767
TIME	Hours from 00 to 24, Minutes from 00 to 59, Seconds from 00 to 62, Milliseconds from 000 to 999
TIMESTAMP	Year from 1 to 9999, Month from 1 to 12, Day from 1 to 28, 29, 30, 31, Hours from 00 to 24, Minutes from 00 to 59, Seconds from 00 to 62, Milliseconds from 000 to 999, Microseconds from 000000 to 999999
VARBINARY	32,705 bytes (256 for an indexed column)
VARCHAR	32,705 characters (256 for an indexed column)

Appendix

C

System parameters for StorHouse/RM

This appendix describes the systems parameters that affect StorHouse/RM and explains how to:

- Display a system parameter value
- Set or change a system parameter value

Refer to the StorHouse *Command Language Reference Manual* for more information about all StorHouse system parameters.

About system parameters

System parameters are named data fields that StorHouse uses to manage resources and to provide default information for system operations. Each parameter has a name and a value. You typically choose values for many system parameters during system installation, but you can change them to configure and tune StorHouse for maximum performance at your site.

System parameter types

There are three types of system parameters:

- A *static* system parameter is one you cannot change while the StorHouse software is operating. Your FileTek customer support representative can assist you should you need to change a static parameter.

- A *dynamic* system parameter is one you can change while the StorHouse software is operating. Changes take effect immediately.
- A *deferred dynamic* system parameter is one you can change while the StorHouse software is operating, but changes do not take effect until after you restart the system.

System parameter access

Some system parameters you can display only, while others you can set and change. The system parameter descriptions starting on page C-4 indicate the type of user access available: SET, SHOW, or SET and SHOW.

- SET indicates you can change the parameter value with the SET SYSTEM command.
- SHOW indicates you can display the parameter value with the SHOW SYSTEM command.
- SET and SHOW indicate you can change and display the parameter value with the SET SYSTEM and SHOW SYSTEM commands.

System parameter list

The following table lists the system parameters that affect StorHouse/RM:

System parameter	Description	Page
LOG_SQL_STMT	User log control for SQL statements	C-4
LOG_SQL_TRANS	User log control for transaction statistics	C-4
SQL_BKUP_ACCOUNT	Metadata backup account	C-5
SQL_BKUP_FSET	Metadata backup file set	C-5
SQL_BKUP_GROUP	Metadata backup file access group	C-5
SQL_BKUP_LIMIT	Metadata backup file limit (maximum)	C-6
SQL_BKUP_VSET	Metadata backup volume set	C-6
SQL_BLD_INDX_MEM	Megabytes of memory for building indexes	C-6
SQL_HOLD_DATA	Days to hold data extents in buffer	C-7
SQL_HOLD_INDX	Days to hold index extents in buffer	C-7
SQL_HOLD_SPECIAL	Days to hold DF and map extents in buffer	C-8
SQL_INDX_TYPE	Default index type	C-8
SQL_LDR_ENGINES	Loader maximum engines	C-9
SQL_LDR_MAXINTO	Loader maximum INTO TABLE clauses	C-9
SQL_LDR_MAXLOAD	Loader maximum LOAD statements	C-10
SQL_MAX_EXT_DATA	Maximum size of a table data extent	C-10
SQL_MAX_EXT_HASH	Maximum size of a hash index data extent	C-11
SQL_MAX_EXT_VAL	Maximum size of a value index data extent	C-11
SQL_SESSIONS	Maximum number of connections	C-12
STORHOUSE_REL	StorHouse/RM software release number	C-12
SYSTEM_ID	Identifier for a StorHouse system	C-12

System parameter descriptions

This section describes the subset of StorHouse system parameters that affect StorHouse/RM.

LOG_SQL_STMT

Enables or disables logging of SQL statements into the user log. TRUE enables logging. FALSE disables logging.

- Expanded Name: User log control for sql_statement records
- Type: Dynamic parameter
- Range: TRUE, FALSE
- Default: TRUE
- User Access: SET, SHOW

LOG_SQL_TRANS

Enables or disables logging of transaction statistics records into the user log. TRUE enables logging. FALSE disables logging.

- Expanded Name: User log control for sql_trans records
- Type: Dynamic parameter
- Range: TRUE, FALSE
- Default: TRUE
- User Access: SET, SHOW

SQL_BKUP_ACCOUNT

Specifies the account that the metadata backup utility uses to write metadata backup files.

- Expanded Name: Metadata backup account
- Type: Dynamic parameter
- Range: Any valid account identifier
- Default: SYSTEM
- User Access: SET, SHOW

SQL_BKUP_FSET

Specifies the file set where the metadata backup utility writes metadata backup files. SQL_BKUP_VSET specifies the volume set.

- Expanded Name: Metadata backup file set
- Type: Dynamic parameter
- Range: Any valid file set name
- Default: RMMDBKUP
- User Access: SET, SHOW

SQL_BKUP_GROUP

Specifies the file access group that the metadata backup utility uses to write metadata backup files.

- Expanded Name: Metadata backup group
- Type: Dynamic parameter
- Range: Any valid file access group name
- Default: RMMDBKUP
- User Access: SET, SHOW

SQL_BKUP_LIMIT

Indicates the maximum number of metadata backup file versions to keep for each database. Based on this value, StorHouse automatically deletes old backup file versions to accommodate new ones. Only the last metadata backup file version is used to recover metadata.

- Expanded Name: Metadata file limit (maximum)
- Type: Dynamic parameter
- Range: 1 through 99
- Default: 10
- User Access: SET, SHOW

SQL_BKUP_VSET

Specifies the volume set where the metadata backup utility writes metadata backup files. SQL_BKUP_FSET specifies the file set.

- Expanded Name: Metadata backup volume set
- Type: Dynamic parameter
- Range: Any valid volume set name
- Default: RMMDBKUP
- User Access: SET, SHOW

SQL_BLD_INDX_MEM

Specifies the number of megabytes of memory that can be used in building indexes during data loads. This memory is used to perform in-memory sorts of blocks of index entries. These blocks are written to disk and later merged to create the index. The larger the memory, the faster the sort but the fewer the loads that can be performed efficiently (without paging) in parallel. The product of this value and SQL_LDR_MAXLOAD should not exceed the virtual memory that can be allocated to data loading processes.

Sorts invoked during user queries will use the value of this parameter to determine the amount of memory used by the sort (the amount used = a minimum of 5 MB or the value of this parameter divided by 10).

- Expanded Name: SQL build index memory
- Type: Dynamic parameter (affects all engines started after the change)
- Range: 10 to 200 MB
- Default: 100
- User Access: SET, SHOW

SQL_HOLD_DATA

Specifies the number of days to hold table data extents in the performance buffer. A value for the HOLD parameter on the CREATE TABLE SPACE or ALTER TABLE SPACE statement overrides this system parameter for a subspace.

Note: Setting values for SQL_HOLD_DATA depends on your data usage patterns. Your FileTek customer support representative can help you set the appropriate value for your site.

- Expanded Name: SQL hold data extents
- Type: Dynamic parameter (affects all engines started after the change)
- Range: 0 to 32767
- Default: 0
- User Access: SET, SHOW

SQL_HOLD_INDX

Specifies the number of days to hold index data extents in the performance buffer. A value for the HOLD parameter on the CREATE TABLE SPACE or ALTER TABLE SPACE statement overrides this system parameter for a subspace.

Note: Setting values for SQL_HOLD_INDX depends on your data usage patterns. Your FileTek customer support representative can help you set the appropriate value for your site.

- Expanded Name: SQL hold index extents
- Type: Dynamic parameter (affects all engines started after the change)
- Range: 0 to 32767
- Default: 0
- User Access: SET, SHOW

SQL_HOLD_SPECIAL

Specifies the number of days to hold DF and index map extents in the performance buffer. A value for the HOLD_SPECIAL parameter on the CREATE TABLE SPACE or ALTER TABLE SPACE statement overrides this system parameter for a subspace.

Note: FileTek recommends that you use very large retention periods for these extents, particularly if you plan to migrate your data to tape.

- Expanded Name: SQL hold special extents
- Type: Dynamic parameter (affects all engines started after the change)
- Range: 0 to 32767
- Default: 0
- User Access: SET, SHOW

SQL_INDX_TYPE

Specifies the default index type for CREATE INDEX statements.

- Expanded Name: SQL index type
- Type: Dynamic parameter (affects all engines started after the change)
- Range: HASH, VALUE, or RANGE
- Default: HASH
- User Access: SET, SHOW

SQL_LDR_ENGINES

Specifies the maximum number of concurrent StorHouse engines used for load operations. One engine is required per LOAD statement. Requests beyond the maximum limit are queued. The SQL_LDR_MAXLOAD system parameter has the same function as SQL_LDR_ENGINES. If the values differ, the smaller value is used.

- Expanded Name: SQL loader maximum engines
- Type: Deferred dynamic parameter (takes effect after system restart)
- Range: 1 through 99
- Default: 25
- User Access: SET, SHOW

SQL_LDR_MAXINTO

Specifies the maximum number of INTO TABLE clauses in any one LOAD statement. If the maximum number of INTO TABLE clauses is exceeded, the load operation fails.

- Expanded Name: SQL loader maximum INTO TABLE clauses
- Type: Deferred dynamic parameter (takes effect after system restart)
- Range: 1 through 99
- Default: 6
- User Access: SET, SHOW

SQL_LDR_MAXLOAD

Specifies the maximum number of LOAD statements that can be processed at a time. One engine is required per LOAD statement. Requests beyond the limit are queued. The SQL_LDR_ENGINES system parameter has the same function as SQL_LDR_MAXLOAD. If the values differ, the smaller value is used.

- Expanded Name: SQL loader maximum LOAD statements
- Type: Deferred dynamic parameter (takes effect after system restart)
- Range: 1 through 99
- Default: 25
- User Access: SET, SHOW

SQL_MAX_EXT_DATA

For table data files, specifies the maximum data extent size (in megabytes) that StorHouse/RM writes to StorHouse. StorHouse/RM checkpoints each data extent when it reaches the maximum size and then creates a new one. StorHouse/RM can recover data extents to the last successful checkpoint. A value for the MAX_EXT_SIZE parameter on CREATE TABLE SPACE or ALTER TABLE SPACE overrides this system parameter for a subspace.

Note: Contact your FileTek customer support representative if you require a data extent size smaller than 100 MB. For a table data file to reach the maximum size (100 GB), set this value to the maximum size (800 MB).

- Expanded Name: SQL maximum extent data
- Type: Dynamic parameter (affects all engines started after the change)
- Range: 100 through 800 MB
- Default: 400
- User Access: SET, SHOW

SQL_MAX_EXT_HASH

For hash index files, specifies the maximum data extent size (in megabytes) that StorHouse/RM writes to StorHouse. StorHouse/RM checkpoints each hash index data extent when it reaches the maximum size and then creates a new one. StorHouse/RM can recover index data extents to the last successful checkpoint. FileTek recommends that you try to contain a hash index in one data extent. Keep in mind, however, that the index must fit on the target device. A value for the MAX_EXT_SIZE parameter on CREATE TABLE SPACE or ALTER TABLE SPACE overrides this system parameter for a subspace.

- Expanded Name: SQL maximum extent hash
- Type: Dynamic parameter (affects all engines started after the change)
- Range: 100 through 800 MB
- Default: 800
- User Access: SET, SHOW

SQL_MAX_EXT_VAL

For value index files, specifies the maximum data extent size (in megabytes) that StorHouse/RM writes to StorHouse. StorHouse/RM checkpoints each value index data extent when it reaches the maximum size and then creates a new one. StorHouse/RM can recover index data extents to the last successful checkpoint. FileTek recommends that you try to contain a value index in one data extent. Keep in mind, however, that the index must fit on the target device. A value for the MAX_EXT_SIZE parameter on CREATE TABLE SPACE or ALTER TABLE SPACE overrides this system parameter for a subspace.

- Expanded Name: SQL maximum extent value
- Type: Dynamic parameter (affects all engines started after the change)
- Range: 100 through 800 MB
- Default: 500
- User Access: SET, SHOW

SQL_SESSIONS

Specifies the maximum number of StorHouse engines that can run concurrently. This sets the maximum number of connections allowed system wide. Note that the maximum number of connections includes the number of loads (one engine per LOAD statement) plus the number of queries. Requests beyond the maximum limit are rejected (the error code is XENORES).

- Expanded Name: SQL maximum number of sessions
- Type: Deferred dynamic parameter (takes effect after system restart)
- Range: 1 through 99
- Default: 50
- User Access: SET, SHOW

STORHOUSE_REL

Specifies the release number of the current StorHouse/RM software in the form x.yz, where x is the version number, y is the release number, and z is a build number. If the value is a null string, StorHouse/RM software is not configured in the system.

- Expanded Name: StorHouse/RM software release number
- Type: Static parameter
- Range: Null string or StorHouse/RM release number
- Default: (Null string)
- User Access: SHOW

SYSTEM_ID

Indicates a a unique system identifier for a StorHouse system. FileTek assigns the value. This identifier is part of a segment file name.

- Expanded Name: StorHouse system identification code
- Type: Static parameter
- Range: 6 decimal characters (value assigned by FileTek)

- Default: Value assigned by FileTek
- Access: SHOW

Displaying a system parameter value

You can display the current value of a StorHouse system parameter by using the StorHouse Command Language SHOW SYSTEM command.

▼ To display a system parameter value

Required privilege: SYSTEM or SHOW

1. Sign on to StorHouse.

2. At the StorHouse command prompt (?), type the following command to display a system parameter value and press J :

   SHOW SYSTEM parameter_name

   where parameter_name is the name of the system parameter. Some examples follow.

   – To display the default index type (value, hash, or range), type:

     SHOW SYSTEM SQL_INDX_TYPE

   – To display the maximum number of LOAD statements that can be processed at a time, type:

     SHOW SYSTEM SQL_LDR_MAXLOAD

Setting or changing a system parameter value

You can set or change the value of a StorHouse system parameter by using the StorHouse Command Language SET SYSTEM command. Depending on the type of system parameter (static, dynamic, dynamic deferred), the change takes effect immediately or after system restart.

▼ To set or change a system parameter value

Required privilege: SYSTEM

1. Sign on to StorHouse.

2. At the StorHouse command prompt (?), type the following command to change a system parameter value and press `J` .

   SET SYSTEM parameter_name value

   where:

   – parameter_name is the name of the system parameter
   – value specifies the new value for the system parameter

   Some examples follow.

   – To change the maximum number of concurrent StorHouse engines used by a FileTek data loader for load operations to 60, type:

     SET SYSTEM SQL_LDR_ENGINES 60

   – To set the number of days to hold special extents in the performance buffer to 365 days, type:

     SET SYSTEM SQL_HOLD_SPECIAL 365

Index

A

access privilege 4-3

access time factor (ATF) 5-7

access to system parameters
- SET C-2
- SET and SHOW C-2
- SHOW C-2

accessing StorHouse from a
- host database application 4-9
- host language application 4-1, 4-10
- locate database interactive tool (like SPUFI) 3-8
- terminal (Interactive Interface) 3-8

account ID
- definition 4-2
- naming conventions 4-2
- on INSERT statement 5-28
- on UPDATE statement 5-29

accounts (StorHouse)
- creating 4-14
- definition 4-1
- disabling 4-14
- general database user 4-8
- loader 4-8
- metadata backup 4-9, 10-3
- password 4-2
- PUBLIC 4-8
- removing 4-27, 4-28
- SQL-enabled 4-8
- SYSADM 4-7
- unloader 4-9

alias, database 2-21

ALL privilege 4-6

ALTER TABLE SPACE statement 5-32

ANSI 1992 Level 2 SQL 1-2, 3-2

application developer, role 3-2

ARCHIVE command 3-4, 10-2

assigning a default user tablespace 5-28

ATF attribute 5-26

B

backing up
- metadata 10-1
- segments 10-2

BINARY data type 6-7

BLOB data type 6-7

C

changing
- default user tablespaces 5-29
- StorHouse access or command privileges 4-21
- StorHouse account passwords 4-20
- StorHouse system parameters C-14

volume and file sets in user tablespaces 5-32

CHARACTER data type 6-8

CHECKPOINT command 10-4

client data loader 1-5

client/server environment feature 1-3

CLOB data type 6-8

column
- definitions 6-5
- names in user tables 6-6
- names in views 8-9
- order in user tables 6-6

column default values
- account ID 6-11
- current date 6-11
- current time 6-11
- literal 6-11
- NULL 6-11

Command Language 3-4

command privileges 4-4

commands (StorHouse)
- ARCHIVE 10-2
- CHECKPOINT 10-4
- CREATE ACCOUNT 4-14
- CREATE BACKUP 10-2
- CREATE FSET 5-23
- CREATE VSET 5-23
- REMOVE ACCOUNT 4-28
- RUN 10-9
- SCHEDULE 10-10
- SET ACCOUNT 4-20, 4-21
- SET SYSTEM 3-6, C-14
- SHOW ACCOUNT 4-22, 4-23
- SHOW FILE 6-21, 7-15
- SHOW FSET 6-21, 7-15
- SHOW SYSTEM 3-6, 5-30, C-13
- SHOW VSET 6-21, 7-15

composite index 7-8

compound index 7-8

concurrent access feature of StorHouse/RM 1-2

Control Center, description xvi

controlled access/database security feature 1-4

conventions, SQL identifier 2-18

CREATE ACCOUNT command 4-14

CREATE BACKUP command 10-2

CREATE FSET command 5-23

CREATE INDEX statement 7-11

CREATE SYNONYM statement 9-4

CREATE VSET command 5-23

creating
- accounts 4-14
- file sets 5-23, 5-24
- indexes for user tables 7-11
- synonyms 9-3
- user tables 6-15
- user tablespaces 5-25
- views from multiple tables 8-7
- volume sets 5-23, 5-24

creator
- user table 6-3
- view 8-3

D

data extent 2-15

data loader feature 1-3

data types
- compression 6-7
- definition 6-6
- limits B-2
- types supported in StorHouse 6-7

data value 6-2

database
- administrator 3-1
- alias 2-21
- component privileges 4-6
- directory 2-5
- integrity 1-4
- local 1-3
- names 2-17
- naming conventions 2-17
- privileges 4-4
- remote 1-3
- system components 2-5
- user components 2-2

DATE data type 6-8

DB2 relational database system 1-2

DBA privilege 4-5

DDL statements 10-2

default definition for a column 6-10

default group 5-10

default index type 7-9

default storage specifications, displaying 5-30

default user tablespace
- assignment priorities 5-19
- definition 5-19
- for a database 5-28
- for an account 5-28
- for an index 5-20, 7-4, 7-5

default values, user tablespace 5-26

DEFAULT VTF value 5-9

deferred dynamic system parameters C-2

definitions
- account 4-1
- ATF 5-7
- client data loader 1-5
- Command Language 3-4
- compound index 7-8
- data extent 2-15
- data type 6-6
- data value 6-2
- database alias 2-21
- database directory 2-5
- database system components 2-5
- database user components 2-2
- default user tablespace 5-19
- deferred dynamic system parameter C-2
- DF extent 2-15
- dynamic system parameter C-2
- EDC 5-9
- extension (SQL) 3-2
- extent 2-15
- extraction 7-7
- field 2-3
- file access group 5-10
- file set (FSET) 5-5
- file set name 5-23
- general database user account 4-5
- hash index 7-4
- hash index file 2-9
- index 7-1
- integrity constraint 6-7
- Interactive Interface 3-8
- local database 1-3
- map extent 2-15

metadata 2-5
metadata backup 10-1
middleware 1-3
not null 6-10
null value 6-9
object identifier 2-13
object type 5-5
out-of-line LOB 2-11
private synonym 2-5
public synonym 2-5
range index 7-5
record 2-3
remote database 1-3
segment 2-7
segment files 2-7
segment tag 2-20
server data loader 1-5
simple index 7-8
static system parameter C-1
StorHouse database 2-1
subspace 5-2
synonym 2-5, 9-1
system parameter 3-5, C-1
system table 2-6, A-1
system table index 2-7
system tablespace 2-5
table data file 2-9
tuple 2-3
undo record 2-7
user log 3-6
user table 2-3, 6-1
user table index 2-4
user tablespace 2-3, 5-1
user tablespace ID 5-21
value index 7-2
value index file 2-9
view 2-4, 8-1
volume set (VSET) 5-4
VTF 5-8

definitions (DF) extent 2-15

DELETE privilege 4-6

delimited SQL identifiers 2-18

devices, storage 1-2

DIRECT VTF value 5-8

DISABLED parameter modifier 4-14

disabling an account 4-14

displaying
- column definitions 6-19
- file set information 6-22, 7-16
- information about indexes 7-12
- information about synonyms 9-5, 9-6
- information about user tables 6-18
- information about user tablespaces 5-34
- information about views 8-10
- list of metadata backup files 10-16
- segment information 6-23, 7-17
- storage specifications for user tablespaces 5-34
- StorHouse system parameters C-13
- system default storage specifications 5-30
- volume set information 6-21, 7-15

DOUBLE PRECISION data type 6-8

DROP INDEX statement 7-19

DROP SYNONYM statement 9-4, 9-7

dropping
- accounts 4-27
- indexes for user tables 7-19
- private synonyms 9-7
- public synonyms 9-7
- user tables 6-25
- user tablespaces 5-35
- view 8-12

dynamic system parameters C-2

E

EDC_COPY system parameter 5-10

EDC_INTERNAL system parameter 5-10

Embedded SQL Interface (ESQL) 1-3

enables 4-4

engine 1-7

error detection code (EDC) 5-9

exporting metadata backup files 10-5

extensions to StorHouse SQL 3-2

extents
- data 2-15
- definition 2-15
- DF 2-15
- map 2-15
- maximum size 5-10
- maximum size defaults 5-11
- retention defaults 5-12
- retention in performance buffer 5-11
- special 5-11

extraction 7-7

extractor 1-3, 7-7

F

features of StorHouse/RM 1-2

field 2-3

file access group 5-10

file names
- segment 2-21
- UNIX 2-26

file naming conventions 2-21

file set name 5-23

File Transfer Protocol (FTP) 1-6

FileTek customer support representative, role 3-2

FileTek FTP Data Loader 1-5

FileTek MVS Data Loader utility 1-4

-FORCE argument (RUN command) 10-9

FSET (file set) 5-5

FTP server, StorHouse 1-5

G

gateway 1-3

general database user account 4-5

GRANT statement 4-11, 4-17

granting
- database component privileges to accounts 4-17
- database privileges to accounts 4-16
- privileges to PUBLIC 4-12

group 5-10

H

hash index file 2-9

hash indexes
- and join processing 7-5
- default user tablespace 7-5
- definition 7-4

how they are stored 7-5
queries that use 7-4

HOLD tablespace parameter 5-26

HOLD_SPECIAL tablespace parameter 5-26

hybrid IN join operation 7-3

I

IDs
account 4-2
segment 2-20
subsegment 2-20
system 2-21
system table 2-26
system table index 2-26
system tablespace 2-26
temporary tablespace 2-26
user table 2-19
user table index 2-20
user tablespace 2-19

index file 2-9

index IDs
conventions 7-10
listing 7-14

index names
conventions 7-9
listing 7-14

INDEX privilege 4-6

indexes
compound 7-8
creating 7-11
definition 2-4, 7-1
displaying information 7-12
dropping 7-19
general rules 7-1
hash 7-4
listing IDs 7-14
listing names 7-14
owners 7-9
range 7-5
simple 7-8
value 7-2

indexing feature 1-3

in-line LOBs 2-11
setting the maximum length 6-12
user tablespace used 6-14

INSERT privilege 4-6

INTEGER data type 6-8

integrity constraint 6-7

Interactive Interface 3-8

issuing
SQL statements 3-8
StorHouse commands 3-8

L

large objects (LOBs)
BLOB data type 6-7
CLOB data type 6-8
INLINE clause 6-12
in-line LOBs 2-11
NOT INLINE clause 6-12
out-of-line LOBs 2-13
setting the maximum length for in-line data 6-12
sharing storage with another LOB column 6-14
storage options 6-12
STORE WITH clause 6-12
storing LOBs in a separate subsegment file 6-13

storing values in a different user tablespace 6-14
subsegment file 2-7
subsegment ID 2-20
TABLESPACE clause 6-12

limits for StorHouse/RM B-1

listing
accounts and their default tablespaces 5-29
database component privileges 4-25
database privileges 4-24
index names and IDs 7-14
metadata backup files 10-16
privileges assigned to PUBLIC 4-23
StorHouse account IDs 4-22
synonym names 9-6
table names and IDs 6-19
user tablespace names and IDs 5-33
view names 8-11

literal column default value 6-11

local database 1-3

LOG_FILE system parameter 3-7

LOG_SQL_STMT system parameter 3-7, C-4

LOG_SQL_TRANS system parameter 3-7

logs
system table 2-7
system table index 2-7
user 3-6

M

map extent 2-15

MAX_EXT_SIZE tablespace parameter 5-26

metadata 2-5, 10-1

metadata backup
backup file naming conventions 10-6
description 10-4
displaying a list of backup files 10-16
exporting backup files 10-5
file retention 10-7
files backed up 10-2
invoking 10-8
messages 10-11
running as needed (RUN) 10-9
scheduling 10-10
system parameters 10-3
what to do after 10-4
when to run 10-2
where files are stored 10-5

middleware 1-3

N

naming a segment 2-20

naming columns in views 8-9

naming conventions
for account IDs 4-2
for account passwords 4-2
for database user components 2-18, 2-19
for databases 2-17
for indexes 7-9
for SQL identifiers 2-18
for subspace numbers 5-21
for synonyms 9-2
for system tables A-2
for user tables 6-4
for user tablespaces 5-20
for views 8-4
in StorHouse 2-16

NEXT VTF value 5-9

not null, definition 6-10

NOW VTF value 5-8

NULL column default value 6-11

null value, definition 6-9

number, subspace 5-21

NUMERIC data type 6-8

O

object identifiers 2-13

object type 5-5

OBJECT_TYPE tablespace parameter 5-26

OIDs 2-13

OPERATOR command privilege 10-3

optimizing SQL 1-7

ordering columns in a user table 6-6

out-of-line LOBs 2-13

owner
- index 7-9
- synonym 9-2
- user table 6-3
- view 8-3

P

package, database A-12

parsing SQL 1-7

password, account 4-2

performance buffer 5-8, 5-11

private synonym
- creating 9-3
- definition 2-5
- dropping 9-7

privileges in StorHouse
- access (SQLADMIN) 4-3
- command
  - SQLCOMMAND 4-4
  - SQLEXECUTE 4-4
- database
  - DBA 4-5
  - RESOURCE 4-5
  - SCAN 4-6
- database component
  - ALL 4-3, 4-6
  - DELETE 4-3, 4-6
  - INDEX 4-6
  - INSERT 4-6
  - SELECT 4-6
  - UPDATE 4-6

PUBLIC
- access to system tables A-3
- considerations 4-12
- default user tablespace 5-28
- definition 4-8

public synonym
- creating 9-4
- definition 2-5
- dropping 9-7

Q

qualified table name 6-3

R

range indexes
- and extractor processing 7-7
- definition 7-5
- how they are stored 7-7
- queries that use 7-6
- storing A-16
- system tables A-16

REAL data type 6-8

record 2-3

remote database 1-3

REMOVE ACCOUNT command 4-28

removing
- accounts 4-27
- indexes 7-19
- private synonyms 9-7
- public synonyms 9-7
- StorHouse access or command privileges 4-21
- StorHouse accounts 4-27, 4-28
- user tables 6-25
- user tablespaces 5-35
- views 8-12

renaming a synonym 9-4

RESOURCE privilege 4-5

restrictions on views 8-4

retrieving data from StorHouse 1-7

revoking
- database component privileges 4-19
- database privileges 4-18

RMMDBKUP volume/file set 10-5

roles for managing StorHouse
- application developer 3-2
- chief database administrator 3-1
- departmental DBA 3-1
- FileTek customer support representative 3-2
- system administrator 3-1

row 2-3

rules for
- SQL identifiers 2-18
- StorHouse databases 2-27

RUN command 10-9

S

SCAN privilege 4-6, 4-9

SCHEDULE command 10-10

segment file 2-7

segment ID 2-20

segment tag 2-20

segments
- backing up 2-16
- definition 2-7
- file names 2-21
- index file 2-9
- invalidation 2-16
- maximum size 2-8, B-2
- naming 2-20

SELECT privilege 4-6, 4-9

SELECT statement
- displaying a column definition 6-20
- displaying information about an index 7-13
- displaying information on tables 6-18
- listing index IDs 7-14
- listing index names 7-14
- listing synonym information 9-5, 9-7

listing table IDs 6-19
listing table names 6-19
unloading data 1-6

server data loader 1-5

SERVICE command privilege 10-3

SET access to system parameters C-2

SET ACCOUNT command 4-20, 4-21

SET and SHOW access to system parameters C-2

SET SYSTEM command 3-6, C-14

setting storage parameters for a user tablespace 5-25

setting up a new user tablespace 5-22

SHOW ACCESS to system parameters C-2

SHOW ACCOUNT command 4-22, 4-23

SHOW FILE /EXTENT command 6-24, 7-18

SHOW FILE command 6-21, 7-15

SHOW FSET command 6-21, 7-15

SHOW SYSTEM command 3-6, 5-30, C-13

simple index 7-8

SMALLINT data type 6-8

software components of StorHouse/RM 1-4

specifying column defaults 6-10

SQL identifiers 2-18

SQL statements, general
description 1-1
features of 1-2
how to submit 3-8
list for database administration 3-3

SQL statements, specific
ALTER TABLE SPACE 5-32
CREATE INDEX 7-11
CREATE SYNONYM 9-4
CREATE TABLE 6-16, 6-17
CREATE TABLE SPACE 5-25, 5-26
CREATE VIEW 8-6
DELETE 6-5
DROP INDEX 7-19
DROP SYNONYM 9-4, 9-7
DROP TABLE 6-15, 6-25
DROP TABLE SPACE 5-35
DROP VIEW 8-12
GRANT 4-11
INSERT 6-5
REVOKE 4-18, 4-19
SELECT 4-24, 6-19, 7-13, 9-5, 9-7
UPDATE 5-29, 6-5

SQL status codes 3-8

SQL_BKUP_ACCOUNT system parameter 10-3, C-5

SQL_BKUP_FSET system parameter 10-3, 10-5, C-5

SQL_BKUP_GROUP system parameter 10-3, C-5

SQL_BKUP_LIMIT system parameter 10-4, 10-7, C-6

SQL_BKUP_VSET system parameter 10-4, 10-5, C-6

SQL_BLD_INDX_MEM system parameter C-6

SQL_HOLD_DATA system parameter 5-12, C-7

SQL_HOLD_INDX system parameter C-7

SQL_HOLD_SPECIAL system parameter C-8

SQL_INDX_TYPE system parameter 7-9, C-8

SQL_LDR_ENGINES system parameter C-9

SQL_LDR_MAXINTO system parameter C-9

SQL_LDR_MAXLOAD system parameter C-10

SQL_MAX_EXT_DATA system parameter 5-11, C-10

SQL_MAX_EXT_HASH system parameter 5-11, C-11

SQL_MAX_EXT_VAL system parameter 5-11, C-11

SQL_SESSIONS system parameter C-12

SQLADMIN privilege 4-3

SQLCOMMAND privilege 4-4

SQLEXECUTE command privilege 4-4

SQLEXECUTE privilege 4-4, 4-9

static system parameters C-1

status codes for SQL statements 3-8

STH default group ID 5-10

storage control attributes
- ATF 5-7
- EDC 5-9
- VTF 5-8

storage devices 1-2

storage management feature 1-4

storage specifications 5-4

StorHouse 1-1
- accounts 4-1, 4-14
- Command Language 3-4
- databases 2-1
- file names 2-21
- file versions 10-7
- FTP server 1-5
- Interactive Interface 3-8
- system administrator 3-1
- system parameters 3-5
- user log 3-6

StorHouse/RM
- engine 1-7
- features 1-2
- position in network 1-2
- software components 1-4

StorHouse/SM 2-1

STORHOUSE_REL system parameter C-12

Structured Query Language (SQL) 1-1

submitting
- SQL statements 3-8
- StorHouse commands 3-8

subsegment files 2-7

subsegment ID 2-20

subspaces
- definition 5-2
- naming 5-21
- selection during loads 5-18

synonyms
- creating 9-3
- definition 2-5, 9-1
- displaying information 9-5, 9-6
- dropping 9-7
- owners 9-2
- private 9-2
- public 9-2
- removing 9-7

SYSADM account 3-9, 4-7

SYSCOLAUTH system table 4-25, A-6

SYSCOLUMNS system table 6-20, A-7

SYSDATE column default value 6-11

SYSDBAUTH system table 4-24, A-8

SYSINDEXES system table 7-14, A-10

SYSPACKAGE system table A-12

SYSPACKSTMT system table A-14

SYSRANGES system tables A-16

SYSSMUSERS system table 5-29, A-19

SYSSTAT_COL system table A-20

SYSSTAT_HIST system table A-21

SYSSTAT_IDX system table A-21

SYSSTAT_SMATRIX system table A-23

SYSSTHFILES system table A-24

SYSSTHSEGMENTS system table A-25

SYSSTHSPACES system table A-26

SYSSYNONYMS system table A-29

SYSTABAUTH system table 4-25, A-30

SYSTABLES system table 6-18, 6-19, A-32

SYSTBLSPACES system table A-34

system administrator, role 3-1

SYSTEM command privilege 10-3

system components, database 2-1

system ID 2-21

system limits B-1

system parameters (StorHouse), general
- changing C-14
- deferred dynamic C-2
- definition C-1
- displaying C-13
- dynamic C-2
- SET access C-2
- SET and SHOW access C-2
- SHOW access C-2
- static C-1

system parameters (StorHouse), specific
- EDC_COPY 5-10
- EDC_INTERNAL 5-10
- LOG_FILE 3-7
- LOG_SQL_STMT 3-7, C-4
- LOG_SQL_TRANS 3-7, C-4
- SQL_BKUP_ACCOUNT 10-3, C-5
- SQL_BKUP_FSET 10-3, C-5
- SQL_BKUP_GROUP 10-3, C-5
- SQL_BKUP_LIMIT 10-4, C-6
- SQL_BKUP_VSET 10-4, C-6
- SQL_BLD_INDX_MEM C-6
- SQL_HOLD_DATA 5-12, C-7
- SQL_HOLD_INDX 5-12, C-7
- SQL_HOLD_SPECIAL 5-12, C-8
- SQL_INDX_TYPE C-8
- SQL_LDR_ENGINES C-9
- SQL_LDR_MAXINTO C-9
- SQL_LDR_MAXLOAD C-10
- SQL_MAX_EXT_DATA 5-11, C-10
- SQL_MAX_EXT_HASH 5-11, C-11
- SQL_MAX_EXT_VAL 5-11, C-11
- SQL_SESSIONS C-12
- STORHOUSE_REL C-12
- SYSTEM_ID C-12

system table index 2-7

system table index log 2-7

system table log 2-7

system tables, general
- definition 2-6, A-1
- list A-4
- name format A-2
- privileges A-3

UNIX file name 2-26

system tables, specific
SYSCOLAUTH 4-25, A-6
SYSCOLUMNS A-7
SYSDBAUTH 4-24, A-8
SYSINDEXES 7-12, A-10
SYSPACKAGE A-12
SYSPACKSTMT A-14
SYSRANGES A-16
SYSSMUSERS 5-29, A-19
SYSSTAT_COL A-20
SYSSTAT_HIST A-21
SYSSTAT_IDX A-21
SYSSTAT_SMATRIX A-23
SYSSTHFILES A-24
SYSSTHSEGMENTS A-25
SYSSTHSPACES A-26
SYSSYNONYMS A-29
SYSTABAUTH 4-25, A-30
SYSTABLES 6-19, A-32
SYSTBLSPACES A-34
SYSVIEWS 8-10, 8-11, A-35

system tablespace 2-5

SYSTEM_ID system parameter C-12

SYSTIME column default value 6-11

SYSVIEWS system table 8-10, 8-11, A-35

T

table data file 2-9

table IDs 6-4

table names 6-3

tables
system A-1
user 6-1

tablespace IDs 5-21

tag, segment 2-20

TIME data type 6-9

TIMESTAMP data type 6-9

transferring data to StorHouse 1-7

tuple 2-3

U

undo record 2-7

UNIX file names 2-26

unloader 1-3, 1-6

UPDATE privilege 4-6

UPDATE statement 5-29

updating default user tablespaces 5-29

USER column default value 6-11

user log 3-6

user tables
column data types 6-6
column definitions 6-5
column order 6-6
creating indexes 7-11
creator 6-3
definition 2-3, 6-1
dropping indexes 7-19
example 2-3
owners 6-3
what's different about StorHouse user tables 6-5

user tablespaces

altering 5-32
creating 5-25
default priorities 5-19, 5-20
default values 5-26
definition 2-3, 5-1
dropping 5-35
examples 5-12
naming conventions 5-20
storage specifications in 5-4
subspaces in 5-2

V

value index file 2-9

value indexes
and join processing 7-3
and segments 7-4
default user tablespace 7-4
how they are stored 7-4
queries that use 7-3

VARBINARY data type 6-9

VARCHAR data type 6-9

versions, StorHouse file 10-7

views
creating 8-6
creating from multiple tables 8-7
creator 8-3
definition 2-4, 8-1
how they work 8-1
naming columns 8-9
owners 8-3
restrictions 8-4

volume set name 5-23

VSET (volume set) 5-4

vulnerability time factor (VTF) 5-8

W

WITH GRANT OPTION 4-11