



StorHouse/RM SQL and Utility Quick Reference Guide

StorHouse/RM Release 3.4

Publication Number
900122 Rev. M

September 17, 2008

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 EXPRESS 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.

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.

NOTICE: U.S. GOVERNMENT USERS

This notice applies to all acquisitions of this work by or for the U.S. Government ("Government"), or by any prime contractor or subcontractor (at any tier) under any contract, cooperative agreement or other activity with the Government. By accepting delivery of this work, the Government agrees that this work and the Licensed Program(s) described herein qualify as "commercial" computer software within the meaning of the acquisition regulation(s) applicable to this procurement. The terms of conditions of the license for the Licensed Program(s) shall pertain to the Government's use and disclosure of this work and the Licensed Program(s), and shall supersede any conflicting contractual terms or conditions. If the license for this work and the Licensed Program(s) fails to meet the Government's need or is inconsistent in any respect with Federal law, the Government agrees to return this work and the Licensed Program(s), unused, to FileTek, Inc. The following additional statement applies only to acquisitions governed by DFARS Subpart 227.4 (October 1988) "Restricted Rights - Use, duplication and disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 (OCT. 1988)." Unpublished licensed work property of FileTek, Inc. Unauthorized use, duplication or distribution prohibited. All rights reserved. A copyright notice on this work and/or on the Licensed Program(s) by itself does not constitute publication or public disclosure of this work or the Licensed Program(s). The contractor/manufacture is:

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

FileTek and StorHouse are registered U.S. trademarks 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. Other patents pending.



StorHouse/RM SQL and Utility Quick Reference Guide

The *StorHouse/RM SQL and Utility Quick Reference Guide* contains formats and examples of the following:

- StorHouse® SQL statements, predicates, and functions
- FileTek data loader LOAD DATA, LOAD INDEX, and MERGE statements
- FileTek data unloader UNLOAD statement
- File Transfer Protocol (FTP) commands to load and unload data
- StorHouse/RM utilities, such as metadata backup, metadata restore, and redo journaling

Conventions

Formats (shown in this font) use the following conventions.

Format conventions

Convention	Description
UPPERCASE	Uppercase terms are keywords that are part of the syntax. Type them as shown.
lowercase	Lowercase terms refer to values that you supply. Quoted strings are case sensitive.
(), / * - ; : . + '	These characters are part of the syntax. Type them as shown.
{ }	Braces indicate the item is required. When a list of items is enclosed in braces and separated by a vertical bar, you must choose one item.
[]	Brackets indicate the item is optional.
	Vertical bars separate alternatives. You can specify one of the alternatives.
...	Ellipsis points indicate you can repeat the part of the statement preceding them any number of times.

Example:

```
CREATE [PUBLIC] SYNONYM synonym_name  
FOR [owner.](table_name | view_name | synonym)
```

Examples (shown in this font) use the following conventions.

Example conventions

Convention	Description
UPPERCASE	Uppercase terms indicate keywords. In non-ESQL statements, uppercase also refers to user-supplied values.
lowercase	Lowercase terms refer to user-supplied values in ESQL (statements that start with EXEC SQL).
;	Semi-colons are required at the end of ESQL statements.

Examples:

```
CREATE PUBLIC SYNONYM SUPPLIERS  
FOR SMITH.SUPPLIERS
```

```
EXEC SQL  
  DECLARE customer_cursor CURSOR FOR  
  SELECT cust_no, state  
  FROM customer ;
```

Conventions

Contents

StorHouse/RM SQL and Utility Quick Reference Guide iii

Conventions iv

Contents vii

SQL statements 1

ALTER TABLE
SPACE 1
BEGIN and
END DECLARE SECTION 1
CLOSE 2
COMMIT WORK 2
CONNECT 2
CREATE EXPLAIN TABLES 3
CREATE INDEX 3
CREATE SYNONYM 3
CREATE TABLE 3
CREATE TABLE SPACE 4
CREATE VIEW 5
DECLARE 5
DELETE 5
DESCRIBE 6
DISCONNECT 6
DROP EXPLAIN TABLES 6

Contents

DROP INDEX	6
DROP SYNONYM	6
DROP TABLE	6
DROP TABLE SPACE	7
DROP VIEW	7
EXECUTE	7
EXECUTE IMMEDIATE	7
EXPLAIN PLAN	7
FETCH	8
FREE LOCATOR	8
GRANT	8
INSERT	9
OPEN	9
PURGE TABLE	9
PREPARE	9
RENAME	9
REVOKE	9
ROLLBACK WORK	10
SELECT	10
SELECT (INTO clause)	11
SELECT (FROM clause)	11
SELECT (WHERE clause)	11
SELECT (GROUP BY clause)	11
SELECT (HAVING clause)	12

SELECT
(ORDER BY clause) 12
SELECT (FOR clause) 12
SET CONNECTION 12
UPDATE 12
VALUES INTO 13
WHENEVER 13

SQL predicates 13

BASIC 13
BETWEEN 14
EXISTS 14
IN 14
LIKE 14
NULL 15
QUANTIFIED 15

SQL functions 15

ABS 15
ADD_MONTHS 16
ASCII 16
AVG 16
BIT_LENGTH 16
BLOB 16
CHAR_LENGTH 16
CHR 17
CLOB 17
CONCAT 17

Contents

COUNT	17
COUNT_BIG	17
DAYOFMONTH	17
DAYOFWEEK	18
DAYOFYEAR	18
DAYS	18
DECODE	18
GREATEST	18
HOURL	19
INITCAP	19
INSTR	19
LAST_DAY	19
LEAST	19
LENGTH	19
LOWER	20
LPAD	20
LTRIM	20
MAX	20
MIN	20
MINUTE	20
MONTH	21
MONTHS_	
BETWEEN	21
NEXT_DAY	21
NVL	21
OCTET_LENGTH	21
OVERLAY	21
POSITION	22
QUARTER	22

RIGHT 22
RPAD 22
RTRIM 22
SECOND 22
SUBSTR 22
SUBSTR_UDB 23
SUM 23
TO_CHAR 23
TO_DATE 23
TO_HEX 23
TO_NUMBER 24
TO_TIME 24
TO_VARCHAR 24
TRANSLATE 24
TRIM 24
UPPER 24
WEEK 25
YEAR 25

Data loader statements 26

LOAD DATA 26
LOAD INDEX 29
MERGE 30

Data unloader statement 31

UNLOAD 31

FTP commands for loading 32

Contents

Commands 32
put keywords and values 33

FTP commands for unloading 35

Commands 35
put keywords and values 36
get keywords and values 36

Utilities 37

sthdb_backup 37
sthdb_down 38
sthdb_restore 38
sthdb_up 38
sthjou_archive 39
sthjou_audit 39
sthjou_cycle 39
sthjou_replay 39
sthseg_delete 40
sthtbl_audit 40
syscreate 40

Limits 41

Index 1

SQL statements

ALTER TABLE SPACE ALTER TABLE SPACE tablespace_name
 (SUBSPACE subspace_number
 param_value [param_value]...
 [,SUBSPACE subspace_number
 param_value [param_value]...]...)

where param_value is any of the following:

[VSET vset_name]
 [FSET fset_name]
 [OBJECT_TYPE T | H | V | L | "" | " "]
 [ATF 0 | 1 | 2 | 3]
 [VTF DEFAULT | NOW | NEXT | DIRECT]
 [EDC D | Y | N]
 [GROUP group_name]
 [MAX_EXT_SIZE size_in_megabytes]
 [HOLD number_of_days]
 [HOLD_SPECIAL number_of_days]

ALTER TABLE SPACE BILLINGTABLE
 (SUBSPACE 1 VSET FEB2000T FSET FEB2000T,
 SUBSPACE 2 HOLD_SPECIAL 65535)

**BEGIN and
 END DECLARE
 SECTION** EXEC SQL BEGIN DECLARE SECTION
 declaration_statements
 EXEC SQL END DECLARE SECTION

where declaration_statements include variable_declarations,
 array_declarations, and type_declarations:

variable_declarations:
 variable_name IS OF TYPE type_category |
 type_category variable_name

SQL statements

```
array_declarations:
variable_name IS AN ARRAY OF type_name
    WITH SIZE constant_id |
host_language_type_name variable_name [constant_id] |
host_language_type_name variable_name [constant_id]
    [length]
```

```
type_declarations:
TYPE new_type_name IS OF TYPE type_category |
TYPE new_type_name IS AN ARRAY OF
    element_type_name WITH SIZE constant_id
```

```
EXEC SQL BEGIN DECLARE SECTION ;
        customer_no IS OF TYPE INTEGER ;
EXEC SQL END DECLARE SECTION ;
```

```
CLOSE EXEC SQL
        CLOSE cursor_name
```

```
EXEC SQL
        CLOSE customer_cursor ;
```

```
COMMIT WORK EXEC SQL
        COMMIT WORK
```

```
EXEC SQL
        COMMIT WORK ;
```

```
CONNECT EXEC SQL
        CONNECT TO database_name
        [ AS connection_name ]
        [ [ USER account_id [USING :host_variable_name ] ] ]
```

```
EXEC SQL
        CONNECT TO 'filetek:T:remotehost:salesdb'
        AS 'conn_2'
        USER 'ssc' USING :sscpwd ;
```

CREATE EXPLAIN TABLES **CREATE EXPLAIN TABLES** [UID identifier]

CREATE EXPLAIN TABLES UID SUE

CREATE INDEX **CREATE** [VALUE | HASH | RANGE] **INDEX** index_name
ON [owner.]table_name (column_name [,column_name]...)
[DEFERRED]
[TABLE SPACE tablespace_name]

CREATE VALUE INDEX ORDINDEX
ON ORDERS (ORDER_NO, CUSTOMER_NAME)
TABLE SPACE ORDERS2000

CREATE SYNONYM **CREATE** [PUBLIC] **SYNONYM** synonym_name
FOR [owner.]{table_name | view_name | synonym}

CREATE PUBLIC SYNONYM SUPPLIERS
FOR SMITH.SUPPLIERS

CREATE TABLE **Format 1: Creating a user table**

CREATE TABLE [owner.]table_name
({column_definition} [,column_definition]...)
[TABLE SPACE tablespace_name]

where column_definition is defined as:

column_name column_type [DEFAULT default_definition]
[column_constraints] [lob_options]

and where lob_options is defined as:

[INLINE [(length [K])] | NOT INLINE]
[STORE WITH column_name]
[TABLE SPACE tablespace_name]

CREATE TABLE SUPPLIER_ITEM
(SUPP_NO INTEGER NOT NULL,
ITEM_NO INTEGER NOT NULL,
QTY INTEGER)

SQL statements

```
CREATE TABLE LOB_EXAMPLE
(CHAR_COL CHAR(6),
INT_COL INTEGER,
FIRST_LOB CLOB STORE WITH OTHER_CLOB,
SECOND_LOB BLOB NOT INLINE,
OTHER_CLOB CLOB TABLESPACE XYZ)
TABLE SPACE ABC
```

Format 2: Undropping a user table

```
CREATE TABLE [[owner.]table_name]
FROM DROPPED [owner.]table_name
[BEFORE timestamp] [INDEX NAMES index [, index]...]
[TABLE SPACE tablespace_name]

CREATE TABLE NEWLOCATION
FROM DROPPED LOCATION
INDEX NAMES NEWLOCATION_IDX1, NEWLOCATION_IDX2
```

CREATE TABLE SPACE CREATE TABLE SPACE tablespace_name
(SUBSPACE subspace_number VSET vset_name FSET
fset_name param_value [param_value]...
[,SUBSPACE subspace_number VSET vset_name FSET
fset_name param_value [param_value]...]...)

where param_value is any of the following:

```
[OBJECT_TYPE T | H | V | L | | "" | " "]
[ATF 0 | 1 | 2 | 3]
[VTF DEFAULT | NOW | NEXT | DIRECT]
[EDC D | Y | N]
[GROUP group_name]
[MAX_EXT_SIZE size_in_megabytes]
[HOLD number_of_days]
[HOLD_SPECIAL number_of_days]

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 1 VTF NEXT MAX_EXT_SIZE 800
HOLD 30 HOLD_SPECIAL 180)

```

CREATE VIEW CREATE VIEW [owner.]view_name
 [(column_name [,column_name]...)]
 AS query_expression

```

CREATE VIEW NE_CUSTOMERS AS
SELECT CUST_NO, NAME, STREET, CITY, STATE, ZIP
FROM CUSTOMER
WHERE STATE IN ('NH', 'MA', 'NY', 'VT')

```

DECLARE EXEC SQL

```

DECLARE cursor_name CURSOR FOR
{ query_expression | prepared_statement_name }

EXEC SQL
DECLARE customer_cursor CURSOR FOR
SELECT cust_no, name, street, city, state
FROM customer ;

```

DELETE DELETE FROM [owner.]{table_name|view_name}
 [WHERE condition]

```

DELETE FROM SYSADM.SYSSMUSERS
WHERE ACCOUNTID='USER1'

```

SQL statements

DESCRIBE EXEC SQL
DESCRIBE BIND VARIABLES FOR statement_name
INTO input_sqlda_pointer

EXEC SQL
DESCRIBE SELECT LIST FOR statement_name
INTO output_sqlda_pointer

EXEC SQL
DESCRIBE BIND VARIABLES FOR dynstmt
INTO isqldaptr ;

EXEC SQL
DESCRIBE SELECT LIST FOR dynstmt
INTO osqldaptr ;

DISCONNECT EXEC SQL
DISCONNECT {connection_name | ALL | CURRENT }

EXEC SQL
DISCONNECT 'conn_2' ;

DROP EXPLAIN TABLES DROP EXPLAIN TABLES [UID identifier]
DROP EXPLAIN TABLES UID SUE

DROP INDEX DROP INDEX index_name [ON [owner.] table name]
DROP INDEX CUSTINDEX ON CUSTOMER

DROP SYNONYM DROP [PUBLIC] SYNONYM synonym
DROP SYNONYM CUSTOMER

DROP TABLE DROP TABLE [owner.]table_name
DROP TABLE CUSTOMER

SQL statements

DROP TABLE SPACE DROP TABLE SPACE tablespace_name
DROP TABLE SPACE BILLINGJAN

DROP VIEW DROP VIEW [owner.]view_name
DROP VIEW NEWCUSTOMERS

EXECUTE EXEC SQL
EXECUTE statement_name
[USING { :host_variable [:indicator_variable]
[:host_variable [:indicator_variable]]...
[DESCRIPTOR input_sqlda_pointer }]

EXEC SQL BEGIN DECLARE SECTION ;
char stmt [256] ;
EXEC SQL END DECLARE SECTION ;
...
EXEC SQL PREPARE stmtid FROM :stmt ;
EXEC SQL EXECUTE stmt USING DESCRIPTOR
sqldaptr ;
...

EXECUTE IMMEDIATE EXEC SQL
EXECUTE IMMEDIATE { :host_variable | statement_string }

EXEC SQL
EXECUTE IMMEDIATE
"DELETE FROM sysadm.syssmusers
WHERE accountid = 'user1' " ;

EXPLAIN PLAN EXPLAIN PLAN [SET STATEMENT_ID='identifier'] FOR query

EXPLAIN PLAN SET STATEMENT_ID='EXAMPLE2'
FOR SELECT * FROM PC JOIN OUTER1 ON
PC.TAGNUM=OUTER1.TAGNUM

SQL statements

```
FETCH EXEC SQL
    FETCH cursor_name
    { INTO :host_variable [:indicator_variable]
      [:host_variable [:indicator_variable] ]...
      | USING DESCRIPTOR output_sqlda_pointer }

...
EXEC SQL
    FETCH cust_cur USING DESCRIPTOR sqldaptr ;
...
```

```
FREE LOCATOR EXEC SQL
    FREE LOCATOR :locator_variable [,:locator_variable]...

EXEC SQL
    FREE LOCATOR :product_locator ;
```

```
GRANT GRANT {RESOURCE, DBA, SCAN}
TO account_id [,account_id ]...

or

GRANT {privilege [,privilege] ... | ALL}
ON {table_name | view_name}
TO {account_id [,account_id ]... | PUBLIC}
[WITH GRANT OPTION]

where privilege is defined as:

{DELETE | INDEX | INSERT | SELECT
 | UPDATE [(column [,column]... )]}

GRANT DBA
TO USER1, USER2

or

GRANT INDEX
ON CUST_VIEW
TO DBUSER1
```

INSERT INSERT INTO [owner.]{table_name | view_name}
 [(column_name [,column_name]...)]
 {VALUES (value [,value]...)
 query_expression}

```
INSERT INTO SYSADM.SYSSMUSERS
(ACCOUNTID, DEFAULT_TS)
VALUES ( 'DBUSER1', 'USER1TS' )
```

OPEN EXEC SQL
 OPEN cursor_name
 [{ USING :host_variable [:indicator_variable]
 [,:host_variable [:indicator_variable]]...
 | USING DESCRIPTOR input_sqlda_pointer }]

```
EXEC SQL
  OPEN ord_cur USING :order_no_v ;
```

PURGE TABLE PURGE TABLE [owner.]table_name [BEFORE droptime]
 PURGE TABLE LOCATION BEFORE '05/30/2006'

PREPARE EXEC SQL
 PREPARE statement_name FROM string_variable
 where string_variable is defined as:
 character_string | :host_variable
 EXEC SQL
 PREPARE delstmt FROM 'delete from
 sysadm.sysismusers where accountid=:mkrl' ;

RENAME RENAME old_object_name TO new_object_name
 RENAME MY.TABLE TO MY.TABLE2

REVOKE REVOKE {RESOURCE, DBA, SCAN}
 FROM account_id [, account_id]...

SQL statements

or

```
REVOKE {privilege [,privilege]... | ALL}
ON {table_name | view_name}
FROM {account_id [,account_id]... | PUBLIC}
```

where privilege is defined as:

```
{DELETE | INDEX | INSERT | SELECT
| UPDATE [(column [,column]... )]}
```

```
REVOKE SCAN
FROM DBUSER1
```

or

```
REVOKE INDEX
ON CUST_VIEW
FROM DBUSER2
```

ROLLBACK WORK EXEC SQL
ROLLBACK WORK

```
EXEC SQL
ROLLBACK WORK ;
```

SELECT SELECT [ALL | DISTINCT]
{* | expr [column_alias] [, expr [column_alias]]...}
[INTO :host_variable [, :host_variable]...]
[FROM table_spec [, table_spec]...]
[WHERE condition]
[GROUP BY column_name [,column_name]... [HAVING
condition]]
[{UNION | UNION ALL} SELECT...]
[ORDER BY {expr | position} [ASC | DESC] [, {expr | position}
[ASC | DESC]]...]
[FOR {FETCH | READ} ONLY]

SELECT INTO :host_variable [, :host_variable]...
(INTO clause)

```
SELECT cust_name
INTO :cust_v
FROM customer
WHERE cust_no=8973205;
```

SELECT FROM table_spec [, table_spec]...
(FROM clause)

where table_spec is:

table_reference [correlation] | joined_table

and where joined_table is:

(joined_table) | table_spec { [INNER] | LEFT [OUTER] }

JOIN table_spec ON join_condition

```
SELECT *
FROM AGENTS
WHERE COMMISSION BETWEEN .10 AND .12

SELECT FIRSTNAME, LASTNAME, PLANE.SERIAL_NUM
FROM PILOT INNER JOIN PLANE
ON PILOT.SERIAL_NUM = PLANE.SERIAL_NUM
LEFT JOIN SERVICE
ON SERVICE.SERIAL_NUM = PILOT.SERIAL_NUM
```

SELECT WHERE condition
(WHERE clause)

```
SELECT *
FROM CUSTOMER
WHERE CITY='BURLINGTON' AND STATE='MA'
```

SELECT GROUP BY column_name [,column_name]...
(GROUP BY clause)

```
SELECT SUBSCRIBER MAX(AMT)
FROM BILLING
GROUP BY SUBSCRIBER
```

SQL statements

SELECT (HAVING clause) HAVING condition

```
SELECT CUST_REP, ODATE, MAX(SALES)
FROM ORDERS
GROUP BY CUST_REP, ODATE
HAVING MAX(SALES) > 5000.00
```

SELECT (ORDER BY clause) ORDER BY {expr | position} [ASC | DESC]
[, {expr | position} [ASC | DESC]]...

```
SELECT NAME, STREET, CITY, STATE, ZIP
FROM CUSTOMER
ORDER BY NAME
```

SELECT (FOR clause) [FOR {FETCH | READ} ONLY]

StorHouse SQL supports the FOR clause for compatibility with DB2[®] SQL only. The clause has no effect on statement execution.

SET CONNECTION EXEC SQL
SET CONNECTION connection_name

```
EXEC SQL
SET CONNECTION 'conn_3' ;
```

UPDATE UPDATE {table_name | view_name}
SET assignment [, assignment]...
[WHERE condition]

where assignment is defined as:

column_name = { expr | NULL } |
(column [, column]...) = (expr [, expr]...)

```
UPDATE SYSADM.SYSSMUSERS
SET DEFAULT_TS = 'DEF99'
WHERE ACCOUNTID = 'BOB'
```



```

VALUES INTO EXEC SQL
    VALUES { expr | (expr [,expr]... ) }
    INTO :host_variable [,host_variable]...

EXEC SQL
    VALUES ( INSTR(:product_locator, 'Product') )
    INTO :start_productinfo;

```

```

WHENEVER EXEC SQL
    WHENEVER exception_sp action_sp

where exception_sp is defined as:
{NOT FOUND | SQLERROR | SQLWARNING}

and action_sp is defined as:
{STOP | CONTINUE | GOTO host_language_label}

EXEC SQL
    WHENEVER SQLERROR GOTO err ;
...
err:
EXEC SQL
    WHENEVER SQLERROR CONTINUE ;

```

SQL predicates

```

BASIC [NOT] basic_pred [AND | OR basic_pred]

where basic_pred is defined as:
expr1 rel_op {expr2 | query_expression}

and rel_op is = , > , < , >= , <= , or <>

SELECT NAME
FROM CUSTOMER
WHERE COUNTRY <> 'USA'

```

SQL predicates

BETWEEN [NOT] between_pred [AND | OR between_pred]

where between_pred is defined as:

expr [NOT] BETWEEN expr1 AND expr2

```
SELECT JOB_TITLE
FROM EMPLOYEE
WHERE SALARY BETWEEN 2000.00 AND 10000.00
```

EXISTS [NOT] exists_pred [AND | OR exists_pred]

where exists_pred is defined as:

EXISTS (query_expression)

```
EXISTS (SELECT * FROM ORDER_TBL
WHERE CUSTID = 4531)
```

IN [NOT] in_pred [AND | OR in_pred]

where in_pred is defined as:

expr [NOT] IN (query_expression) | (:host_variable | literal)
[,(:host_variable | literal)]...

```
ADDRESS.STATE IN ( 'MA' , 'NH' )
```

LIKE [NOT] like_pred [AND | OR like_pred]

where like_pred is defined as:

column_name [NOT] LIKE {string_constant | :host_variable}
[ESCAPE character_constant1]

```
CUST_NAME LIKE "%Computer%"
```

NULL [NOT] null_pred [AND | OR null_pred]

where null_pred is defined as:

column_name IS [NOT] NULL

```
SELECT ORDER_NO
FROM ORDERS
WHERE CONTACT_NAME IS NULL
```

QUANTIFIED [NOT] quant_pred [AND | OR quant_pred]

where quant_pred is defined as:

expr rel_op {ANY | SOME} (query_expression)

and rel_op is = , > , < , >= , <= , or <>

```
SELECT NAME
FROM CUSTOMER
WHERE COUNTRY = ANY
(SELECT COUNTRY
FROM SUPPLIER)
```

SQL functions

ABS ABS (expression)

```
SELECT ABS (MONTHS_BETWEEN (SYSDATE,
ORDER_DATE))
FROM ORDERS
WHERE ABS (MONTHS_BETWEEN (SYSDATE,
ORDER_DATE)) > 3
```

SQL functions

ADD_MONTHS ADD_MONTHS (date_expression, integer_expression)

```
SELECT *
FROM CUSTOMER
WHERE ADD_MONTHS (START_DATE, 6) > SYSDATE
```

ASCII ASCII (char_expression)

```
SELECT ASCII (ZIP)
FROM CUSTOMER
```

AVG AVG ({[ALL] expression} | {DISTINCT column_ref})

```
SELECT AVG (SALARY)
FROM EMPLOYEE
WHERE DEPTNO = 20
```

BIT_LENGTH BIT_LENGTH (expression)

```
SELECT BIT_LENGTH(NAME)
FROM CUSTOMER;
```

BLOB BLOB (expression [, length])

```
SELECT *
FROM BLOB_DATA
WHERE BLOB_COLUMN = BLOB('12345678')
```

CHAR_LENGTH CHAR_LENGTH (expression)

```
SELET NAME
FROM CUSTOMER
WHERE CHAR_LENGTH(NAME) > 10
```

CHR CHR (integer_expression)

```
SELECT *  
FROM CUSTOMER  
WHERE SUBSTR (ZIP, 1, 1) = CHR (53)
```

CLOB CLOB (char_expression [, length])

```
SELECT CLOB(NAME)  
FROM CUSTOMER
```

CONCAT CONCAT (expression1, expression2)

```
SELECT NAME, EMPNO, SALARY  
FROM CUSTOMER  
WHERE PROJECT = CONCAT('US', PROJ_NAM)
```

COUNT COUNT ([{ALL} expression] | [DISTINCT] column_ref | *)

```
SELECT COUNT (DISTINCT ORDER_NUM)  
FROM ORDERS
```

COUNT_BIG COUNT_BIG ([{ALL} expression] | [DISTINCT] column_ref | *)

```
SELECT COUNT_BIG (*)  
FROM ORDERS  
WHERE ORDER_DATE = SYSDATE
```

DAYOFMONTH DAYOFMONTH (date_expression)

```
SELECT *  
FROM ORDERS  
WHERE DAYOFMONTH (ORDER_DATE) = 14
```

SQL functions

DAYOFWEEK DAYOFWEEK (date_expression)

```
SELECT *
FROM ORDERS
WHERE DAYOFWEEK (ORDER_DATE) = 2
```

DAYOFYEAR DAYOFYEAR (date_expression)

```
SELECT *
FROM ORDERS
WHERE DAYOFYEAR (ORDER_DATE) = 300
```

DAYS DAYS (date_expression)

```
DAYS('0001-01-01')
DAYS('0001-01-02')
DAYS('1998-09-08')
```

DECODE DECODE (expression, search_expression, match_expression
[, search_expression, match_expression]...
[, default_expression])

```
SELECT ENAME,
DECODE (DEPTNO,
10, 'ACCOUNTS',
20, 'RESEARCH',
30, 'SALES',
40, 'SUPPORT',
'NOT ASSIGNED')
FROM EMPLOYEE
```

GREATEST GREATEST (expression [,expression]...)

```
SELECT CUST_NO, NAME,
GREATEST (LAST_PYMT_DATE, LAST_CALL_DATE)
FROM CUSTOMER
```

HOUR HOUR (time_expression)

```
SELECT *  
FROM ARRIVALS  
WHERE HOUR (IN_TIME) < 12
```

INITCAP INITCAP (char_expression)

```
SELECT INITCAP (NAME)  
FROM CUSTOMER
```

INSTR INSTR (expression1, expression2
[,start_position [,occurrence]])

```
SELECT CUST_NO, NAME  
FROM CUSTOMER  
WHERE INSTR (ADDR, 'heritage') > 0
```

LAST_DAY LAST_DAY (date_expression)

```
SELECT *  
FROM ORDERS  
WHERE LAST_DAY (ORDER_DATE) = '08/31/1995'
```

LEAST LEAST (expression [,expression]...)

```
SELECT CUST_NO, NAME,  
LEAST (LAST_PYMT_DATE, LAST_CALL_DATE)  
FROM CUSTOMER
```

LENGTH LENGTH (expression)

```
SELECT NAME 'LONG NAME'  
FROM CUSTOMER  
WHERE LENGTH (NAME) > 5
```

SQL functions

LOWER LOWER (char_expression)

```
SELECT NAME
FROM CUSTOMER
WHERE LOWER (NAME) = 'smith'
```

LPAD LPAD (char_expression, length [, pad_expression])

```
SELECT LPAD (NAME, 30)
FROM CUSTOMER
```

LTRIM LTRIM (expression, char_set)

```
SELECT NAME, LTRIM (ADDR, ' ')
FROM CUSTOMER
```

MAX MAX (expression)

```
SELECT ORDER_DATE, PRODUCT, MAX (QTY)
FROM ORDERS
GROUP BY ORDER_DATE, PRODUCT
```

MIN MIN (expression)

```
SELECT MIN (SALARY)
FROM EMPLOYEE
WHERE DEPTNO = 20
```

MINUTE MINUTE (time_expression)

```
SELECT *
FROM ARRIVALS
WHERE MINUTE (IN_TIME) > 10
```


MONTH MONTH (date_expression)

```
SELECT *  
FROM ORDERS  
WHERE MONTH (ORDER_DATE) = 6
```

MONTHS_ MONTHS_BETWEEN (date_expression1, date_expression2)
BETWEEN

```
SELECT MONTHS_BETWEEN (SYSDATE, ORDER_DATE)  
FROM ORDERS  
WHERE ORDER_NO = 1002
```

NEXT_DAY NEXT_DAY (date_expression, day_of_week)

```
SELECT NEXT_DAY (ORDER_DATE, 'MONDAY')  
FROM ORDERS
```

NVL NVL (expression1, expression2)

```
SELECT SALARY + NVL (COMM, 0) 'TOTAL SALARY'  
FROM EMPLOYEE
```

OCTET_LENGTH OCTET_LENGTH (expression)

```
SELECT OCTET_LENGTH(NAME)  
FROM CUSTOMER
```

OVERLAY OVERLAY (expression1 PLACING expression2
FROM start_position [FOR length])

```
SELECT NAME, OVERLAY(PHONE PLACING '858'  
FROM 1)  
FROM CUSTOMER  
WHERE SUBSTR(PHONE,1,3) = '619'
```

SQL functions

POSITION POSITION (expression1 IN expression2)

```
SELECT CUST_NO, NAME
FROM CUSTOMER
WHERE INSTR(ADDR, 'heritage') > 0
```

QUARTER QUARTER (date_expression)

```
SELECT *
FROM ORDERS
WHERE QUARTER (ORDER_DATE) = 3
```

RIGHT RIGHT (expression, length)

```
SELECT ACCTNUM (RIGHT,10)
FROM CUSTOMER
```

RPAD RPAD (char_expression, length [, pad_expression])

```
SELECT RPAD (NAME, 30, '.')
FROM CUSTOMER
```

RTRIM RTRIM (expression, char_set)

```
SELECT RPAD (RTRIM (ADDR, ' '), 30, '.')
FROM CUSTOMER
```

SECOND SECOND (time_expression)

```
SELECT *
FROM ARRIVALS
WHERE SECOND (IN_TIME) <= 40
```

SUBSTR SUBSTR (expression, start_position [, length])

```
SELECT NAME, '(', SUBSTR (PHONE, 1, 3) , ')',
SUBSTR (PHONE, 4, 3), '-',
```

```
SUBSTR (PHONE, 7, 4)
FROM CUSTOMER
```

or

```
SUBSTR ( expression FROM start_position [FOR length] )

SELECT NAME, '(' , SUBSTR(PHONE FROM 1 FOR 3) ,
' ) ' ,
SUBSTR (PHONE FROM 4 FOR 3) , ' - ' ,
SUBSTR (PHONE FROM 7 FOR 4)
FROM CUSTOMER
```

SUBSTR_UDB SUBSTR_UDB (expression, start_position [, length])

```
SELECT NAME SUBSTR (CUSTID, 1, 10)
FROM CUSTOMER
```

SUM SUM ({[ALL] expression} | {DISTINCT column_ref})

```
SELECT SUM (AMOUNT)
FROM ORDERS
WHERE ORDER_DATE = SYSDATE
```

TO_CHAR TO_CHAR (expression [,format])

```
SELECT NAME, TO_CHAR (START_DATE,
'DD-MON-YYYY' )
FROM CUSTOMER
```

TO_DATE TO_DATE (char_expression [,format])

```
SELECT *
FROM ORDERS
WHERE ORDER_DATE <= TO_DATE ('12/31/1991')
```

TO_HEX TO_HEX (expression)

SQL functions

```
SELECT TO_HEX(PHONE)
FROM TABLE1
```

TO_NUMBER TO_NUMBER (char_expression)

```
SELECT *
FROM CUSTOMER
WHERE TO_NUMBER (SUBSTR (PHONE, 1, 3)) = 603
```

TO_TIME TO_TIME (char_expression [,format])

```
SELECT *
FROM ORDERS
WHERE ORDER_DATE < TO_DATE ('05/15/1991')
AND ORDER_TIME < TO_TIME ('12:00:00')
```

TO_VARCHAR TO_VARCHAR (expression [,maxlength])

```
SELECT TO_VARCHAR (NAME, 32) FROM CUSTOMER
```

TRANSLATE TRANSLATE (char_expression, from_set, to_set)

```
SELECT NAME, TRANSLATE (STREET, ' ', '_')
FROM CUSTOMER
```

TRIM TRIM ([[LEADING | TRAILING | BOTH] [expression1]
FROM] expression2)

```
SELECT NAME, TRIM(BOTH ' ' FROM ADDR)
FROM CUSTOMER
```

UPPER UPPER (char_expression)

```
SELECT *
FROM CUSTOMER
WHERE UPPER (NAME) = 'SMITH'
```

WEEK WEEK (date_expression)

```
SELECT *  
FROM ORDERS  
WHERE WEEK (ORDER_DATE) = 5
```

YEAR YEAR (date_expression)

```
SELECT *  
FROM ORDERS  
WHERE YEAR (ORDER_DATE) = 1999
```

Data loader statements

```
LOAD DATA LOAD [ DATA ]  
[ CHARACTERSET { cset_name | ccsid } ]  
[ { INFILE | INDDN } { [ NOENVIRON ] (infile_list) | * | - } ]  
[ load_options ]  
{ into_table_spec }...
```

Argument	Format
cset_name	WE8EBCDIC500 WE8PC850 WE8ISO8859P1
ccsid	500 850 819
(infile_list)	sm_file_name [/group] [NOENVIRON] [, sm_file_name [/group] [NOENVIRON]]...
load_options	[{ DISCARDFILE DISCARDN } sth_file_name [/group]] [{ DISCARDS DISCARDMAX } num_discards] [CONCATENATE num_lines] [CONTINUEIF continueif_condition] [PRESERVE BLANKS] [SUBSPACE ROTATE] [ESCAPED BY [DELIMITER 'char' NONE]]
continueif_condition	{ [THIS NEXT] (position) LAST } operator { any_string BLANKS }
position	start_column [{ : - } end_column]
operator	= != ^= <> < > <= >=
any_string	string Xstring

Data loader statements

Argument	Format
into_table_ spec	INTO TABLE [owner.] table_name [WHEN field_condition [{ AND OR } field_condition]...] [FIELDS fields_specs] [TRAILING [NULLCOLS]] [SAME SEGMENT] [DIFF[ERENT] SEGMENT] [SEGMENT segment_tag] [REPLACE SEGMENT [[owner.] table_name.] segment_tag] [[TABLE VALUE HASH LOB] SUBSPACE number]... [(field_spec [,field_spec]...)]
field_ condition	{ (position) column_name field_name } operator { any_string BLANKS }
fields_ specs	[CHAR] [NULLFLAGS] [delimiter_spec] [NULLIF (EMPTY BLANK)] [DEFAULTIF (EMPTY BLANK)]
delimiter_ spec	[TERMINATED [BY] { WHITESPACE 'char' X'hexbyte' }] [[OPTIONALLY] ENCLOSED [BY] { 'char' X'hexbyte' }] [AND { 'char' X'hexbyte' }]]
field_spec	{ :field_name column_name } data_spec

Data loader statements

Argument	Format
data_spec	RECNUM SEQUENCE (start_num [,increment]) SYSDATE SYSTIME SYSTIMESTAMP CONSTANT any_value position_spec
any_value	any_string identifier n
position_spec	[POSITION (position * [+num])] [datatype_spec] [NULLIF field_condition] [DEFAULTIF field_condition]
datatype_spec	Refer to the <i>FileTek FTP Data Loader Manual</i> or the <i>FileTek MVS Data Loader Manual</i>

load data into table instreamlobs
fields nullflags;

or

```
LOAD DATA
INTO TABLE CALLSDBA.BILLSUMMARY
TABLE SUBSPACE 1
HASH SUBSPACE 2
VALUE SUBSPACE 3
(BILL_ACCOUNT BINARY EXTERNAL(10),
BILL_DATE DATE EXTERNAL(10),
BILL_CATEGORY CHAR(1),
NUMBER_CALLS INT EXTERNAL(3))
WHEN BILL_CATEGORY='1'
```


Data loader statements

```

INTO TABLE CALLSDBA.BILLSUMMARY
DIFFERENT SEGMENT
TABLE SUBSPACE 4
HASH SUBSPACE 5
VALUE SUBSPACE 6
(BILL_ACCOUNT POSITION(1) BINARY EXTERNAL(10),
BILL_DATE DATE EXTERNAL(10),
BILL_CATEGORY CHAR(1),
NUMBER_CALLS INT EXTERNAL(3))
WHEN BILL_CATEGORY='2';

```

```

BEGINDATA;
301792833901/31/20001004,
703274028301/31/20001002,
703872283901/31/20002001,
301339439201/31/20001003,
301340920301/31/20001002,
703419408301/31/20002003,
703229383901/31/20002001,

```

LOAD INDEX LOAD INDEX[ES] {index_name,... | [index_name,...]
 FOR TABLE [owner.] tablename} [subspace_clause]
 [SEGMENTS segment_list]]

Argument	Format
subspace_clause	SUBSPACE ROTATE [VALUE HASH] SUBSPACE number
segment_list	segment_list_item [, segment_list_item]...
segment_list_item	segment_range segment
segment_range	first_segment - last_segment

Data loader statements

```
LOAD INDEX ORDERS2000 SEGMENTS 0-5
SUBSPACE ROTATE
```

or

```
LOAD INDEX ORDERS2000
```

MERGE {MERGE | COALESCE} INTO TABLE table_name
[subspace_clause] [SEGMENT segment_tag]
[SEGMENTS segment_list] [EXCLUDE segment_list]
[MAXINSIZE n] [MINOUTSIZE n]

Argument	Format
subspace_clause	SUBSPACE ROTATE [VALUE HASH TABLE] SUBSPACE number
segment_list	IDS segment_list_item [, segment_list_item]... TAGS segment_tag [, segment_tag]...
segment_list_item	segment_range segment
segment_range	first_segment - last_segment

```
MERGE INTO TABLE CALLSDETAIL SEGMENTS IDS 0-4
or
MERGE INTO TABLE CALLSDETAIL MAXINSIZE 100M
MINOUTSIZE 1G EXCLUDE TAGS "late_entries"
```

Data unloader statement

```
UNLOAD UNLOAD
[unload_options]
[ USING ( field_spec [ , field_spec ]... )
query ;
```

where unload_options is any of the following clauses in any order:

```
[ CHARACTERSET { cset_name | ccid } ]
[ OUTFILE { sth_file_name } [ /group ] ]
[ FIELDS [ CHAR ] [ NULLFLAGS ] delimiter_spec ]
[ ESCAPED BY [ DELIMITER | 'char' | NONE ] ]
[ RECORDS TERMINATED [ BY ] { WHITESPACE | 'char' |
X'hexbyte' } ]
[RECORDS NOT TERMINATED]
```

```
UNLOAD
FIELDS NULLFLAGS
SELECT ITEM, PHOTO FROM INVENTORY;
```

or

```
UNLOAD
FIELDS TERMINATED BY ' ','
RECORDS TERMINATED BY ';'
USING (CONSTANT 'insert into otbl values (',
CHAR ENCLOSED BY '"' IFNULL='NULL',',
INTEGER EXTERNAL IFNULL='NULL',',
CHAR ENCLOSED BY '"' IFNULL='NULL',
CONSTANT ')' )
SELECT ITEM, QUANTITY, STATUS FROM INVENTORY;
```

FTP commands for loading

Commands	FTP command	Description
	ascii	Sets the transfer type to ASCII. You can also use the type ascii command.
	binary	Sets the transfer type to BINARY. You can also use the type image command.
	bye	Logs off the StorHouse FTP server and quits FTP.
	close	Logs off the StorHouse FTP server but doesn't quit FTP.
	open	Logs into the StorHouse FTP server at alternate port 1985.
	put or send	Transfers the control file and the data file or pipes the input from a program; confirms, restarts, aborts, and checks the status of an operation.
	quit	Logs off the StorHouse FTP server and quits FTP.
	remotehelp	Provides a list of server-level FTP commands, or when followed by a command name, a definition for a specific command.
	type ascii	Sets the transfer type to ASCII.
	type image	Sets the transfer type to BINARY.

FTP commands for loading

put keywords and values {put | send} local-file keyword=value[,keyword=value]...

Keyword	Values
[load_type=]	<ul style="list-style-type: none"> ■ load ■ data ■ confirm or end ■ restart ■ status ■ abort
db_ref=	<ul style="list-style-type: none"> ■ ANSI (default) ■ Oracle ■ DB2
sql_ccsid=	<ul style="list-style-type: none"> ■ 819 for ISO 8859-1 (default) ■ 500 for EBCDIC ■ 850 for PC
data_ccsid=	<ul style="list-style-type: none"> ■ 819 for ISO 8859-1 (default) ■ 500 for EBCDIC ■ 850 for PC
dbn[ame]=	Database name
fixed=	Fixed-length data record length (in bytes)
var	Variable-length data indicator
loadid[ent]=	Load ID
nvk=	<ul style="list-style-type: none"> ■ SPARC (default) ■ IBM ■ DOS ■ VAX

FTP commands for loading

Transfer a control file:

```
ftp> put control.inp  
load,dbn=database1,loadid=1
```

Transfer a data file:

```
ftp> put data.inp data,fixed=50
```

Restart a load:

```
ftp> put control.inp  
restart,dbn=database1,loadid=1  
  
ftp> put data.inp data,fixed=50
```

Confirm a load:

```
ftp> put - confirm
```

Abort a load:

```
ftp> put control.inp  
abort,dbn=database1,loadid=1
```

FTP commands for unloading

Commands

FTP command	Description
ascii	Sets the transfer type to ASCII. You can also use the type ascii command.
binary	Sets the transfer type to BINARY. You can also use the type image command.
bye	Logs off the StorHouse FTP server and quits FTP.
close	Logs off the StorHouse FTP server but doesn't quit FTP.
get	Retrieves the result data, placing it in a file on your computer or piping it to another program.
open	Logs into the StorHouse FTP server at alternate port 1985.
put	Starts an unload and transfers the control file.
quit	Logs off the StorHouse FTP server and quits FTP.
remotehelp	Provides a list of server-level FTP commands, or when followed by a command name, a definition for a specific command.
type ascii	Sets the transfer type to ASCII.
type image	Sets the transfer type to BINARY.

FTP commands for unloading

put keywords and values put local-file keyword=value[,keyword=value]...

Keyword	Values
[load_type=]	unload
dbn[ame]=	Database name
sql_ccsid=	<ul style="list-style-type: none">■ 819 for ISO 8859-1 (default)■ 500 for EBCDIC■ 850 for PC
data_ccsid=	<ul style="list-style-type: none">■ 819 for ISO 8859-1 (default)■ 500 for EBCDIC■ 850 for PC
fixed=	Fixed-length data record length (in bytes)
var	Variable-length binary record indicator
nvk=	<ul style="list-style-type: none">■ SPARC (default)■ IBM■ DOS■ VAX

get keywords and values get keyword=value[,keyword=value]... local-file

Keyword	Values
[load_type=]	data
fixed=	Record length of fixed-length records
var	Variable-length record indicator

Log in to the StorHouse FTP server:

```
ftp alpha1 1985
```

or

```
ftp> open alpha1 1985
```

Set the transfer type to binary:

```
ftp> type i
```

Transfer a control file:

```
ftp> put control.inp  
load_type=unload,dbn=db1
```

Get a result file:

```
ftp> get load_type=data,fixed=32767 data..out
```

Log off the StorHouse FTP server, quit FTP:

```
ftp> bye
```

Utilities

Command option letters may be uppercase or lowercase, for example, -f or -F.

sthdb_backup sthdb_backup [options] database_name [database_name]...

[options]

- f (force run)
- h (help)
- j (start journaling)
- r (reset journaling)
- n (non-interactive mode)
- v (verbose)
- s {sm_options}

Utilities

where {sm_options} are:

VSET=name,FSET=name,VTF=NOW | DIRECT | NEXT

sthdb_backup -f callsdb INVENTORY

sthdb_backup calls -v

-s VSET=MYVSET,FSET=MYFSET,VTF=NOW

sthdb_down sthdb_down [options] database_name

[options]

-n (non-interactive mode)

-h (help)

sthdb_down callsdb

sthdb_restore sthdb_restore [options] database_name[:~n]
[-a alternate_database_name] ...

[options]

-r (read verification)

-n (non-interactive mode)

-v (verbose)

-h (help)

sthdb_restore callsdb -r callsdb1

sthdb_up sthdb_up [options] database_name

[options]

-f (force up)

-n (non-interactive mode)

-h (help)

sthdb_up callsdb

sthjou_archive sthj jou_archive [options] database_name

[options]

- p N[.nnnnn](purge disk journal files)
 - o (purge only, no archiving)
 - c N (number of safe copies that must exist)
- s {sm_options}

where {sm_options} are:

VSET=name,FSET=name,VTF=NOW | DIRECT | NEXT

```
sthjou_archive -s vset=v2006,fset=f2006
callsdb
```

sthjou_audit sthj jou_audit [options] database_name[:n]

[options]

- f (detailed audit)
- l N (limit number of journal files audited)
- h (help)

```
sthjou_audit -f callsdb:-2
```

sthjou_cycle sthj jou_cycle [options] database_name

[options]

- v (verbose)
- h (help)

```
sthjou_cycle -v callsdb
```

sthjou_replay sthj jou_replay [options] database_name

[options]

- d N[.nnnnn] (partial replay, days)
- l (list journal files)
- v (verbose)
- h (help)

```
sthjou_replay -d 2 -v callsdb
```

Utilities

sthseg_delete sthseg_delete [options] database_name

[options]

-d days (days invalid)

-t tblid[,tblid]... (table IDs)

-b days (days dropped)

-p (purge tables first)

-l lock_seconds (system table lock limit)

-v verbose_level (messaging level, 0 through 3)

sthseg_delete -d 30 -p callsdb

sthtbl_audit sthtbl_audit [options] dbname tblid

[options]

-d all | deleted | valid | tid[:tid] Dump selected records

-f Fix any problems in the table

-n Do not obtain a lock (on the table)

-s Do a full table scan and associated checks/stats

dbname Database name

tblid TableID

sthtbl_audit -p 10000 -s TESTBUILD34 2

syscreate syscreate [options] database_name

[options]

-j (enable journaling)

syscreate -j callsdb

Limits

Item	Limit
Length of a database name	32 characters
Length of a database component name	32 characters
Length of data types	
BIGINT	-9223372036854775808 (-2 ⁶³) to +9223372036854775807 (+2 ⁶³ -1)
BINARY	256 bytes
BLOB	2 ³¹ -9
CHARACTER	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

Limits

Item	Limit
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)
Number of columns in a compound index	16 columns
Number of columns in a table	1,024 columns
Number of indexed columns for a table	2,400 columns
Size of a column in an index	256 bytes
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	2,147,483,638 bytes

Item	Limit
Size of a key in a compound index	4,096 bytes
Size of a table data file	100 gigabytes
Size of an SQL statement	32,767 characters

Limits

Index

A

ABS scalar function 15

ADD_MONTHS scalar function 16

aggregate functions

AVG 16

COUNT 17

COUNT_BIG 17

MAX 20

MIN 20

SUM 23

ALTER TABLE SPACE statement 1

ascii FTP command 32, 35

ASCII scalar function 16

AVG aggregate function 16

B

basic predicate 13

BEGIN DECLARE SECTION statement 1

BETWEEN predicate 14

binary FTP command 32, 35

BIT_LENGTH scalar function 16

BLOB scalar function 16

bye FTP command 32, 35

C

- CHAR_LENGTH scalar function 16
- CHR scalar function 17
- CLOB scalar function 17
- close 32
- close FTP command 32, 35
- CLOSE statement 2
- commands, FTP
 - ascii 32, 35
 - binary 32, 35
 - bye 32, 35
 - close 32, 35
 - get 35, 36
 - open 32, 35
 - put 32, 35
 - quit 32, 35
 - remotehelp 32, 35
 - type ascii 32, 35
 - type image 32, 35
- COMMIT WORK statement 2
- CONCAT scalar function 17
- CONNECT statement 2
- CONNECTION statement 12
- COUNT aggregate function 17
- COUNT_BIG aggregate function 17
- CREATE EXPLAIN TABLES statement 3
- CREATE INDEX statement 3
- CREATE SYNONYM statement 3
- CREATE TABLE SPACE statement 4

CREATE TABLE statement 3

CREATE VIEW statement 5

D

data types 41

DAYOFMONTH scalar function 17

DAYOFWEEK scalar function 18

DAYOFYEAR scalar function 18

DAYS scalar function 18

db_ref put keyword 33

dbname put keyword 33, 36

DECLARE statement 5

DECODE scalar function 18

DELETE statement 5

DESCRIBE statement 6

DISCONNECT statement 6

DROP EXPLAIN TABLES statement 6

DROP INDEX statement 6

DROP SYNONYM statement 6

DROP TABLE statement 6, 7

DROP VIEW statement 7

E

END DECLARE SECTION statement 1

EXECUTE 7

EXECUTE IMMEDIATE statement 7

F

EXECUTE statement 7

EXISTS predicate 14

EXPLAIN PLAN statement 7

F

FETCH statement 8

fixed get keyword 36

fixed put keyword 33, 36

FOR clause 12

FREE LOCATOR statement 8

FROM clause 11

G

get FTP command 35

GRANT statement 8

GREATEST scalar function 18

GROUP BY clause 11

H

HAVING clause 12

HOURLY scalar function 19

I

IN predicate 14

INITCAP scalar function 19

INSERT statement 9

INSTR scalar function 19

INTO clause 11

K

keywords, get command

- fixed 36

- load_type 36

- var 36

keywords, put command

- data_ccsid 33, 36

- db_ref 33

- dbname 33, 36

- fixed 33, 36

- load_type 33, 36

- loadident 33

- nvk 33, 36

- sql_ccsid 33, 36

- var 33, 36

L

LAST_DAY scalar function 19

LEAST scalar function 19

LENGTH scalar function 19

LIKE predicate 14

limits 41

LOAD DATA statement 26

LOAD INDEX statement 29

M

- load_type get keyword 36
- load_type put keyword 33, 36
- loadident put keyword 33
- LOWER scalar function 20
- LPAD scalar function 20
- LTRIM scalar function 20

M

- MAX aggregate function 20
- MERGE statement 30
- MIN aggregate function 20
- MINUTE scalar function 20
- MONTH scalar function 21
- MONTHS_BETWEEN scalar function 21

N

- NEXT_DAY scalar function 21
- NULL predicate 15
- nvk put keyword 33, 36
- NVL scalar function 21

O

- OCTET_LENGTH scalar function 21
- open FTP command 32, 35
- OPEN statement 9

ORDER BY clause 12

OVERLAY scalar function 21

P

POSITION scalar function 22

predicates

 basic 13

 BETWEEN 14

 EXISTS 14

 IN 14

 LIKE 14

 NULL 15

 quantified 15

PREPARE statement 9

PURGE TABLE statement 9

put FTP command 32, 35

Q

quantified predicate 15

QUARTER scalar function 22

quit FTP command 32, 35

R

remotehelp FTP command 32, 35

RENAME statement 9

REVOKE statement 9

RIGHT scalar function 22

ROLLBACK WORK statement 10

RPAD scalar function 22

RTRIM scalar function 22

S

scalar functions

ABS 15

ADD_MONTHS 16

ASCII 16

BIT_LENGTH 16

BLOB 16

CHAR_LENGTH 16

CHR 17

CLOB 17

CONCAT 17

DAYOFMONTH 17

DAYOFWEEK 18

DAYOFYEAR 18

DAYS 18

DECODE 18

GREATEST 18

HOUR 19

INITCAP 19

INSTR 19

LAST_DAY 19

LEAST 19

LENGTH 19

LOWER 20

LPAD 20

LTRIM 20

MINUTE 20

MONTH 21

MONTHS_BETWEEN 21

NEXT_DAY 21

NVL 21

OCTET_LENGTH 21

OVERLAY 21
POSITION 22
QUARTER 22
RIGHT 22
RPAD 22
RTRIM 22
SECOND 22
SUBSTR 22
SUBSTR_UDB 23
TO_CHAR 23
TO_DATE 23
TO_HEX 23
TO_NUMBER 24
TO_TIME 24
TO_VARCHAR 24
TRANSLATE 24
TRIM 24
UPPER 24
WEEK 25
YEAR 25

SECOND scalar function 22

SELECT (FOR clause) statement 12

SELECT (FROM clause) statement 11

SELECT (GROUP BY clause) statement 11

SELECT (HAVING clause) statement 12

SELECT (INTO clause) statement 11

SELECT (ORDER BY clause) statement 12

SELECT (WHERE clause) statement 11

SELECT statement 10

sql_ccsid put keyword 33, 36

statements

ALTER TABLE SPACE 1

BEGIN DECLARE SECTION 1

CLOSE 2

COMMIT WORK 2
CONNECT 2
CONNECTION 12
CREATE EXPLAIN TABLES 3
CREATE INDEX 3
CREATE SYNONYM 3
CREATE TABLE 3
CREATE TABLE SPACE 4
CREATE VIEW 5
DECLARE 5
DELETE 5
DESCRIBE 6
DISCONNECT 6
DROP EXPLAIN TABLES 6
DROP INDEX 6
DROP SYNONYM 6
DROP TABLE 6, 7
DROP VIEW 7
END DECLARE SECTION 1
EXECUTE 7
EXECUTE IMMEDIATE 7
EXPLAIN PLAN 7
FETCH 8
FREE LOCATOR 8
GRANT 8
INSERT 9
LOAD DATA 26
LOAD INDEX 29
MERGE 30
OPEN 9
PREPARE 9
PURGE TABLE 9
RENAME 9
REVOKE 9
ROLLBACK WORK 10
SELECT 10
SELECT (FOR clause) 12
SELECT (FROM clause) 11
SELECT (GROUP BY clause) 11

- SELECT (HAVING clause) 12
- SELECT (ORDER BY clause) 12
- SELECT (WHERE clause) 11
- UNLOAD 31
- UPDATE 12
- VALUES INTO 13
- WHENEVER 13
- sthdb_backup utility 37
- sthdb_down utility 38
- sthdb_restore utility 38
- sthdb_up utility 38
- sthjou_archive utility 39
- sthjou_cycle utility 39
- sthjou_replay utility 39
- sthseg_delete utility 40
- sthtbl_audit utility 40
- SUBSTR scalar function 22
- SUBSTR_UDB scalar function 23
- SUM aggregate function 23
- syscreate utility 40

T

- tables
 - LOAD DATA format 26
 - LOAD INDEX format 29
 - MERGE format 30
- TO_CHAR scalar function 23
- TO_DATE scalar function 23
- TO_HEX scalar function 23

U

- TO_NUMBER scalar function 24
- TO_TIME scalar function 24
- TO_VARCHAR scalar function 24
- TRANSLATE scalar function 24
- TRIM scalar function 24
- type ascii FTP command 32, 35
- type image FTP command 32, 35

U

- UNLOAD statement 31
- UPDATE statement 12
- UPPER scalar function 24
- utilities
 - sthdb_backup 37
 - sthdb_down 38
 - sthdb_restore 38
 - sthdb_up 38
 - sthjou_archive 39
 - sthjou_cycle 39
 - sthjou_replay 39
 - sthseg_delete 40
 - sthtbl_audit 40
 - syscreate 40

V

- VALUES INTO statement 13
- var get keyword 36
- var put keyword 33, 36

W

WEEK scalar function 25

WHENEVER statement 13

WHERE clause 11

Y

YEAR scalar function 25