



StorHouse SQL Quick Reference

StorHouse/RM Release 3.2

Publication Number
900122 Rev. K

April 8, 2002

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.

Note to the US Government: Restricted Rights Legend. Use, duplication, or disclosure by the Government is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013. The contractor/manufacturer is FileTek, Inc., 9400 Key West Avenue, Rockville, MD 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.



Copyright © 1997-2002 by FileTek, Inc., Rockville, MD
Publication Number: 900122 Rev. K

StorHouse SQL Quick Reference

This Quick Reference contains formats and examples of StorHouse® SQL statements, predicates, and functions. It serves as a mini-reference for the *StorHouse SQL Reference Manual*, which contains complete information about StorHouse SQL.

Conventions

SQL formats (shown in this font) use the following conventions.

SQL 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)
```

SQL examples (shown in this font) use the following conventions.

SQL 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 ;
```


Contents

StorHouse SQL statements

ALTER TABLE SPACE	1
BEGIN and END DECLARE SECTION	1
CLOSE	2
COMMIT WORK	2
CONNECT	2
CREATE INDEX	3
CREATE SYNONYM	3
CREATE TABLE	3
CREATE TABLE SPACE	4
CREATE VIEW	5
DECLARE	5
DELETE	5
DESCRIBE	5
DISCONNECT	6
DROP INDEX	6
DROP SYNONYM	6
DROP TABLE	6
DROP TABLE SPACE	6
DROP VIEW	6
EXECUTE	6
EXECUTE IMMEDIATE	7
FETCH	7
FREE LOCATOR	7
GRANT	8
INSERT	8

OPEN	8
PREPARE	9
REVOKE	9
ROLLBACK WORK	9
SELECT	10
SELECT (INTO clause)	10
SELECT (FROM clause)	10
SELECT (WHERE clause)	11
SELECT (GROUP BY clause)	11
SELECT (HAVING clause)	11
SELECT (ORDER BY clause)	11
SELECT (FOR clause)	11
SET CONNECTION	11
UPDATE	12
VALUES INTO	12
WHENEVER	12

StorHouse SQL predicates

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

StorHouse SQL functions

ABS	15
ADD_MONTHS	15
ASCII	15
AVG	15
BIT_LENGTH	15
BLOB	16
CHAR_LENGTH	16
CHR	16
CLOB	16
CONCAT	16
COUNT	17
DAYOFMONTH	17
DAYOFWEEK	17
DAYOFYEAR	17
DAYS	17
DECODE	18
GREATEST	18
HOUR	18
INITCAP	18
INSTR	18
LAST_DAY	19
LEAST	19
LENGTH	19
LOWER	19
LPAD	19
LTRIM	19
MAX	20

MIN	20
MINUTE	20
MONTH	20
MONTHS_BETWEEN	20
NEXT_DAY	20
NVL	21
OCTET_LENGTH	21
OVERLAY	21
POSITION	21
QUARTER	21
RPAD	21
RTRIM	22
SECOND	22
SUBSTR	22
SUM	22
TO_CHAR	22
TO_DATE	23
TO_HEX	23
TO_NUMBER	23
TO_TIME	23
TRANSLATE	23
TRIM	23
UPPER	24
WEEK	24
YEAR	24

Limits 25

Index

StorHouse 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 | blank
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

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 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 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)
```

```
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
```

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 | blank
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'

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 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

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'" ;
```

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

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.syssmusers where accountid=:mkr1'  
;
```

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

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 clause) WHERE condition

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

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

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

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 ;
```

StorHouse 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'
```

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)
```


StorHouse SQL functions

ABS ABS (expression)

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

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(X'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 ( * )  
FROM ORDERS  
WHERE ORDER_DATE = SYSDATE
```

DAYOFMONTH DAYOFMONTH (date_expression)

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

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
```

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_BETWEEN MONTHS_BETWEEN (date_expression1,
date_expression2)

```
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'
```

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
```

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
```

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)

```
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')
```

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
```

Limits

Item	Limit
Length of a database name	32 characters (16 for DRDA)
Length of a database component name	32 characters
Length of data types	
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	-32768 to 32767

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	32705 bytes (256 for an indexed column)
VARCHAR	32705 characters (256 for an indexed column)
Number of columns in a compound index	16 columns
Number of columns in a table	500 columns
Number of indexes for a table	150 indexes
Size of a column in an index	256 bytes
Size of a row in a table	32,705 bytes

Item	Limit
Size of an in-line LOB	32,705 bytes
Size of an out-of-line LOB	2,147,483,638 bytes
Size of a key in a compound index	2048 bytes
Size of a table data file	100 gigabytes
Size of an SQL statement	10,000 bytes

Index

A

ABS scalar function 15

ADD_MONTHS scalar function 15

aggregate functions

 AVG 15

 COUNT 17

 MAX 20

 MIN 20

 SUM 22

ALTER TABLE SPACE statement 1

ASCII scalar function 15

AVG aggregate function 15

B

basic predicate 13

BEGIN DECLARE SECTION statement 1

BETWEEN predicate 13

BIT_LENGTH scalar function 15

BLOB scalar function 16

C

CHAR_LENGTH scalar function 16

- CHR scalar function 16
- CLOB scalar function 16
- CLOSE statement 2
- COMMIT WORK statement 2
- CONCAT scalar function 16
- CONNECT statement 2
- CONNECTION statement 11
- COUNT aggregate function 17
- CREATE INDEX statement 3
- CREATE SYNONYM statement 3
- CREATE TABLE SPACE statement 4
- CREATE TABLE statement 3
- CREATE VIEW statement 5

D

- data types 25
- DAYOFMONTH scalar function 17
- DAYOFWEEK scalar function 17
- DAYOFYEAR scalar function 17
- DAYS scalar function 17
- DECLARE statement 5
- DECODE scalar function 18
- DELETE statement 5
- DESCRIBE statement 5
- DISCONNECT statement 6
- DROP INDEX statement 6

DROP SYNONYM statement 6

DROP TABLE statement 6

DROP VIEW statement 6

E

END DECLARE SECTION statement 1

EXECUTE 6

EXECUTE IMMEDIATE statement 7

EXECUTE statement 6

EXISTS predicate 13

F

FETCH statement 7

FOR clause 11

FREE LOCATOR statement 7

FROM clause 10

G

GRANT statement 8

GREATEST scalar function 18

GROUP BY clause 11

H

HAVING clause 11

HOUR scalar function 18

I

IN predicate 13

INITCAP scalar function 18

INSERT statement 8

INSTR scalar function 18

INTO clause 10

L

LAST_DAY scalar function 19

LEAST scalar function 19

LENGTH scalar function 19

LIKE predicate 14

limits 25

LOWER scalar function 19

LPAD scalar function 19

LTRIM scalar function 19

M

MAX aggregate function 20

MIN aggregate function 20

MINUTE scalar function 20

MONTH scalar function 20

MONTHS_BETWEEN scalar function 20

N

NEXT_DAY scalar function 20

NULL predicate 14

NVL scalar function 21

O

OCTET_LENGTH scalar function 21

OPEN statement 8

ORDER BY clause 11

OVERLAY scalar function 21

P

POSITION scalar function 21

predicates

- basic 13

- BETWEEN 13

- EXISTS 13

- IN 13

- LIKE 14

- NULL 14

- quantified 14

PREPARE statement 9

Q

quantified predicate 14

QUARTER scalar function 21

R

REVOKE statement 9

ROLLBACK WORK statement 9

RPAD scalar function 21

RTRIM scalar function 22

S

scalar functions

ABS 15

ADD_MONTHS 15

ASCII 15

BIT_LENGTH 15

BLOB 16

CHAR_LENGTH 16

CHR 16

CLOB 16

CONCAT 16

DAYOFMONTH 17

DAYOFWEEK 17

DAYOFYEAR 17

DAYS 17

DECODE 18

GREATEST 18

HOUR 18

INITCAP 18

INSTR 18

LAST_DAY 19

LEAST 19

LENGTH 19

LOWER 19

LPAD 19

LTRIM 19

MINUTE 20

MONTH 20
MONTHS_BETWEEN 20
NEXT_DAY 20
NVL 21
OCTET_LENGTH 21
OVERLAY 21
POSITION 21
QUARTER 21
RPAD 21
RTRIM 22
SECOND 22
SUBSTR 22
TO_CHAR 22
TO_DATE 23
TO_HEX 23
TO_NUMBER 23
TO_TIME 23
TRANSLATE 23
TRIM 23
UPPER 24
WEEK 24
YEAR 24

SECOND scalar function 22

SELECT (FOR clause) statement 11

SELECT (FROM clause) statement 10

SELECT (GROUP BY clause) statement 11

SELECT (HAVING clause) statement 11

SELECT (INTO clause) statement 10

SELECT (ORDER BY clause) statement 11

SELECT (WHERE clause) statement 11

SELECT statement 10

statements

ALTER TABLE SPACE 1

BEGIN DECLARE SECTION 1

CLOSE 2
COMMIT WORK 2
CONNECT 2
CONNECTION 11
CREATE INDEX 3
CREATE SYNONYM 3
CREATE TABLE 3
CREATE TABLE SPACE 4
CREATE VIEW 5
DECLARE 5
DELETE 5
DESCRIBE 5
DISCONNECT 6
DROP INDEX 6
DROP SYNONYM 6
DROP TABLE 6
DROP VIEW 6
END DECLARE SECTION 2
EXECUTE 6
EXECUTE IMMEDIATE 7
FETCH 7
FREE LOCATOR 7
GRANT 8
INSERT 8
OPEN 8
PREPARE 9
REVOKE 9
ROLLBACK WORK 9
SELECT 10
SELECT (FOR clause) 11
SELECT (FROM clause) 10
SELECT (GROUP BY clause) 11
SELECT (HAVING clause) 11
SELECT (ORDER BY clause) 11
SELECT (WHERE clause) 11
UPDATE 12
VALUES INTO 12
WHENEVER 12

SUBSTR scalar function 22

SUM aggregate function 22

T

TO_CHAR scalar function 22

TO_DATE scalar function 23

TO_HEX scalar function 23

TO_NUMBER scalar function 23

TO_TIME scalar function 23

TRANSLATE scalar function 23

TRIM scalar function 23

U

UPDATE statement 12

UPPER scalar function 24

V

VALUES INTO statement 12

W

WEEK scalar function 24

WHENEVER statement 12

WHERE clause 11

Y

YEAR scalar function 24