

SW-Tools ODBC - Programmers Reference

22/11/01 / 2022-09-01 008.384

# Contents

Contents 2

1. Preface 3

2. Installation 4

3. Principle of operation 5

4. ODBC.INI parameters 6

5. Functions 7

6. Examples of varius use of SQL in the SW-Tools ODBC driver 8

6.1. Calculations 10

6.2. Special column names 11

6.3. Selecting using the IN clause 12

6.4. Correlation names 13

6.5. OUTER JOIN 14

6.6. Subqueries 15

6.7. Aggregate functions 17

6.8. LIKE and MATCHES 18

6.9. How to use functions within SELECT statements 19

6.10. Date, Time and timestamp 20

6.11. Using field numbers 21

6.12. Field subscriptions 22

6.13. GROUP BY, HAVING, DISTINCT and UNION 23

6.14. VALUES constructor and SELECT from result set 25

7. Updating the database and datadictionary itself 26

7.1. Copying table INTO TEMP 27

7.2. INSERT values INTO table 28

7.3. INSERT values from other tables 29

7.4. Updating existing records 32

7.5. DELETE multiple records 34

7.6. GRANT/REVOKE implimentation 35

7.7. DROP table 36

8. Current of cursors 37

8.1. Getting CURSOR name 39

8.2. Setting CURSOR name 40

8.3. SELECT for UPDATE 41

8.4. SELECT from cursor 42

8.5. DELETE from cursor 43

9. Views 44

10. Create / Alter and Rename tables 45

10.1. How to create tables 46

10.2. ALTER table definition 47

11. Data types 48

12. Table types, names, Owners and Qualifiers 52

13. Parameters 55

14. Parameters - Data at execution. 56

15. Options 57

16. Functions 59

17. SQLInfo 60

Figure list 68

Index 69

# 1. Preface

SW-Tools ODBC driver is compliant to ODBC 2.10, API-Level 1, SQL core level.

Most of the extended SQL instruction set is implemented as shown below.

The driver is delivered in 32 bit version only.

# 2. Installation

The driver is installed using the SETUP program on the CD.

By use of the ODBC Administrator setup function you can define multiple data sources to be used with the driver.

# 3. Principle of operation

The SW-Tools ODBC driver uses the TRIO Data Dictionary to access files using SQL on any implemented file system supported by SW-Tools.

This opens access to a lot of Windows products as ACCESS, EXCEL, WORD etc.

This short example collection is intended for programmmers reference merely as user handbook - the end user should focus on the application programs only.

# 4. ODBC.INI parameters

The following is a complete list of possible entries in ODBC.INI

Me= Default path the drivers files

Basis= Path for BASIS.SSV defining the file system interfaces

Dmf= Path for the datadictionary FILES.SSV and xx.SSD

Isa= Default path for the database files if needed

Com= Company number

Based= Normally blank, forces all files to a given BASIS filetype

Fixfil= 0 Forces the driver to read FILES.SSV whenever accessed

Upper= 0 Use upper/lowercase names instead of just uppercase

Fname= 0 Use File ID only as SQL tablenames

Fnamelen= n Use max n characters in tablename length

Ftext 0 Usage of file text desctiption

Qualifier= 0 Return NULL instead of file ID as table qualifier

Owner= 0 Return NULL instead of filetype as table owner

Lan= ENG The language is fixed on the disk

Test= 1 Internal testflags producing a c:\wif testoutput

Update= 1 Data source is not readonly, requires full release

# 5. Functions

The following is a list of implemented functions, refer first to the ODBC manual SQL functions then to the SW-Tools TRIO calculations and subfunctions manual.

ABS, ACOS, ASCII, ASIN, ATAN, ATAN2, CEILING, CHAR, CONCAT, CONV, COS, COT, CURDATE, CURTIME, DATABASE, DATE, DAY, DAYNAME, DAYOFMONTH, DAYOFWEEK, DAYOFYEAR, DEGREES, EDIT, EXP, FIND, FLOOR, FNA, FNB, FND, FNE, FNF, FNH, FNO, FNR, FNU, FNV, FNY, FRA, HOUR, IN, INSERT, INT, ISNULL, LCASE, LEFT, LEN, LENGTH, LIKE, LOCATE, LOG, LOG10, LOWER, LTRIM, MATCHES, MINUTE, MOD, MONTH, MONTHNAME, NAME, NOT, NOW, NUMBER, NUMS, PI, POW, POWER, QUARTER, RADIANS, RAND, REPEAT, REPLACE, RIGTH, ROUND, RTRIM, RUN, RUND, SECOND, SGN, SIGN, SIN, SMAA, SOGE, SPACE, SPOFF, SQR, SQRT, SUBSTRING, TAN, TIME, TOCHAR, TODBL, TOLONG, TOSHORT, TRUNCATE, UCASE, UPPER, USER, VALCH, VALID, WDAY, WEEK, YEAR

# 6. Examples of varius use of SQL in the SW-Tools ODBC driver

Simple SQL statements examples with access of one table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 2 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 3 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 4 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |
| 5 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |
| 6 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 7 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |

1. Simple SELECT

SELECT \*

FROM va

ORDER BY may reference any column, field and DESC/ASC may be used.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Address** | **Town** | **Curre** | **Balance** |
| 1 | 271 | DANDY INC. | 13-MAIN STREET | LOS ANGELES | 2 | 0 |
| 2 | 270 | OHIO INC. | MAIN AVENUE | NEW YORK | 2 | 200 |
| 3 | 123 | BRAUN GMBH | PLATZ DAMN 12 | LUXEMBOURG | 1 | 0 |
| 4 | 205 | SCHIERMACHER LTD. | BOULEVARD ROYAL 63 | LUXEMBOURG | 1 | 20000 |
| 5 | 100 | HUMBER LTD. | HUMBER STREET 223 | 4711 COPENHAGEN S | 0 | 0 |
| 6 | 105 | WEBB'S SUPPLIERS LTD. | EAST STREET 373 | 4711 COPENHAGEN F | 0 | 500 |
| 7 | 111 | TRAWSOM LTD. | WEST STREET 111 | 1820 COPENHAGEN C | 0 | 1000 |
| 8 | 260 | CLORID LTD. | COPENHAGEN STREET 3 | 1154 COPENHAGEN K | 0 | 2000 |
| 9 | 102 | AX & AX LTD. | SEA PARK ROAD 43 | 2100 COPENHAGEN | 0 | 25000 |

2. Using ORDER BY

SELECT \*

FROM le

ORDER BY 5 DESC,balance

Basic files may be handled just as other database tables

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Recty** | **Debtor/c** | **Name 1** | **Name 2** | **Street** | **Town** | **Country** | **Postal code** | **FC co** | **Gros** | **W** |
| 1 | 1 | 10000 | Otto Mühlmeier | Einzelhändler | Richard-Wurzbacher-Str. 12 | Moers | >T02315485160 | 47441 | 0 | 0 | 1 |
| 2 | 1 | 10001 | Berliner Handels KG | Großhandel | Ostdorfer Straße 48 | Berlin | >T02315485160 | 10111 | 0 | 1 | 1 |
| 3 | 1 | 10002 | Hans von der Kooij | Großhandel | Europlaan 101 | Utrecht | >T02315485160 | NL-3200 DJ | 5 | 1 | 2 |
| 4 | 1 | 10003 | Hatch Ltd. |  | 300, Third Avenue | Waltham | >T02315485160 | MA 02154 | 3 | 1 | 1 |
| 5 | 1 | 10004 | Megatrent KG | Großhandel | Friedrich-Ebert-Straße 123-127 | Duisburg | >T02315485160 | 47163 | 0 | 1 | 1 |
| 6 | 1 | 12001 | Bathen Filiale 1 |  |  |  |  |  | 0 | 0 | 0 |
| 7 | 1 | 69999 | Diverse Debitoren |  |  |  | >T02315485160 |  | 0 | 1 | 1 |

3. ODBC on BASIC files with simple WHERE clause

SELECT \*

FROM GF-03000

WHERE rectype=1

# 6.1. Calculations

Calculations may be performed both for columns and in WHERE If result columns are not named they becomes the name EXPR-1,2,...

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **EXPR-1** | **calc** | **new** |
| 1 | 100 | HUMBER LTD. | 0 | 2 | 9 |
| 2 | 102 | AX & AX LTD. | 175000 | 25002 | 25009 |
| 3 | 123 | BRAUN GMBH | 0 | 2 | 9 |
| 4 | 205 | SCHIERMACHER LTD. | 140000 | 20002 | 20009 |
| 5 | 271 | DANDY INC. | 0 | 2 | 9 |

4. Using calculations

SELECT no, name, balance \* 7, balance + 2 calc, calc + 7 new

FROM le

WHERE new+1 NOT BETWEEN 207 + 1 AND 2999

# 6.2. Special column names

Special column names must be enclosed in '...', optionally for legal names

|  |  |  |  |
| --- | --- | --- | --- |
|  | **No** | **Balance** | **Name** |
| 1 | 102 | 25000 | AX & AX LTD. |
| 2 | 205 | 20000 | SCHIERMACHER LTD. |
| 3 | 260 | 2000 | CLORID LTD. |

5. Special column names encloded in quotes

SELECT no,'balance','le'.'name'

FROM 'le'

WHERE 'balance'>1000

The IN function may be used to select records:

# 6.3. Selecting using the IN clause

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 2 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |

SELECT \*

FROM va

WHERE supplier IN ("123","271")

# 6.4. Correlation names

Several files can be used (joined) in one select. Correlation names (AS a) for tables may be given, AS may be omitted. The correlation name does not have to be given if no duplicate column names exists.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Price** | **No** | **Suppli** | **Altern** | **Name** |
| 1 | 100 | 20000 | 0102 | 100 | 0 | HUMBER LTD. |
| 2 | 100 | 2000 | 1005 | 100 | 0 | HUMBER LTD. |
| 3 | 100 | 100000 | 0110 | 123 | 100 | HUMBER LTD. |
| 4 | 102 | 1000 | 1001 | 205 | 102 | AX & AX LTD. |
| 5 | 102 | 25 | 2001 | 205 | 102 | AX & AX LTD. |
| 6 | 102 | 25 | 2002 | 205 | 102 | AX & AX LTD. |
| 7 | 123 | 100000 | 0110 | 123 | 100 | BRAUN GMBH |
| 8 | 205 | 1000 | 1001 | 205 | 102 | SCHIERMACHER LTD. |
| 9 | 205 | 25 | 2001 | 205 | 102 | SCHIERMACHER LTD. |
| 10 | 205 | 25 | 2002 | 205 | 102 | SCHIERMACHER LTD. |
| 11 | 270 | 2 | 0101 | 271 | 270 | OHIO INC. |
| 12 | 271 | 2 | 0101 | 271 | 270 | DANDY INC. |

6. Using AS clause for correlation name of table

SELECT no, price, a.no, supplier, alternative, name

FROM le, va AS a

WHERE no=supplier OR no=alternative

# 6.5. OUTER JOIN

Tables may be joined using the OUTER JOIN facility, below also suppliers without articles are in the result set.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Price** | **No** | **Suppli** | **Altern** | **Name** |
| 1 | 100 | 20000 | 0102 | 100 | 0 | HUMBER LTD. |
| 2 | 100 | 2000 | 1005 | 100 | 0 | HUMBER LTD. |
| 3 | 102 | 0 |  |  | 0 | AX & AX LTD. |
| 4 | 105 | 0 |  |  | 0 | WEBB'S SUPPLIERS LTD. |
| 5 | 111 | 0 |  |  | 0 | TRAWSOM LTD. |
| 6 | 123 | 100000 | 0110 | 123 | 100 | BRAUN GMBH |
| 7 | 205 | 1000 | 1001 | 205 | 102 | SCHIERMACHER LTD. |
| 8 | 205 | 25 | 2001 | 205 | 102 | SCHIERMACHER LTD. |
| 9 | 205 | 25 | 2002 | 205 | 102 | SCHIERMACHER LTD. |
| 10 | 260 | 0 |  |  | 0 | CLORID LTD. |
| 11 | 270 | 0 |  |  | 0 | OHIO INC. |
| 12 | 271 | 2 | 0101 | 271 | 270 | DANDY INC. |

7. Using OUTER JOIN

SELECT no, price, a.no, supplier, alternative, le.name

FROM le, OUTER va a

WHERE supplier=no

The full ODBC extended escape clause for outer joins are supported, however *only* **LEFT OUTER JOINS** are implemented.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Price** | **No** | **Suppli** | **Altern** | **Name** |
| 1 | 205 | 1000 | 1001 | 205 | 102 | SCHIERMACHER LTD. |
| 2 | 205 | 25 | 2001 | 205 | 102 | SCHIERMACHER LTD. |
| 3 | 205 | 25 | 2002 | 205 | 102 | SCHIERMACHER LTD. |
| 4 | 260 | 0 |  |  | 0 | CLORID LTD. |
| 5 | 270 | 0 |  |  | 0 | OHIO INC. |
| 6 | 271 | 2 | 0101 | 271 | 270 | DANDY INC. |

8. Using LEFT OUTER JOIN

SELECT no, price, a.no, supplier, alternative, le.name FROM

{ oj le LEFT OUTER JOIN va a ON supplier=no }

WHERE no>200

# 6.6. Subqueries

Subqueries can be performed.

|  |  |  |
| --- | --- | --- |
|  | **No** | **Name** |
| 1 | 100 | HUMBER LTD. |
| 2 | 123 | BRAUN GMBH |
| 3 | 205 | SCHIERMACHER LTD. |
| 4 | 271 | DANDY INC. |

9. Multiple SELECTs for subqueries

SELECT no,name

FROM le a

WHERE EXISTS

(SELECT \* FROM va WHERE supplier=a.no)

Comparision operators may be used for subqueries

|  |  |  |  |
| --- | --- | --- | --- |
|  | **No** | **Balance** | **Name** |
| 1 | 102 | 25000 | AX & AX LTD. |
| 2 | 105 | 500 | WEBB'S SUPPLIERS LTD. |
| 3 | 111 | 1000 | TRAWSOM LTD. |
| 4 | 205 | 20000 | SCHIERMACHER LTD. |
| 5 | 260 | 2000 | CLORID LTD. |
| 6 | 270 | 200 | OHIO INC. |

10. Using comparision operators

SELECT no,balance,name

FROM le a

WHERE balance > ALL

(SELECT price+7 FROM va WHERE supplier=a.no)

EXISTS, ALL, ANY, SOME may be used.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **No** | **Balance** | **Name** |
| 1 | 100 | 0 | HUMBER LTD. |
| 2 | 123 | 0 | BRAUN GMBH |
| 3 | 271 | 0 | DANDY INC. |

11. Sample use of ANY comparision

SELECT no,balance,name

FROM le a

WHERE balance < ANY

(SELECT price+7 FROM va WHERE supplier=a.no)

By use of IN a result set may be scanned for values

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Address** | **Town** | **Curre** | **Balance** |
| 1 | 100 | HUMBER LTD. | HUMBER STREET 223 | 4711 COPENHAGEN S | 0 | 0 |
| 2 | 123 | BRAUN GMBH | PLATZ DAMN 12 | LUXEMBOURG | 1 | 0 |
| 3 | 205 | SCHIERMACHER LTD. | BOULEVARD ROYAL 63 | LUXEMBOURG | 1 | 20000 |
| 4 | 271 | DANDY INC. | 13-MAIN STREET | LOS ANGELES | 2 | 0 |

12. Scanning result set when using IN clause

SELECT \*

FROM le

WHERE no IN (SELECT supplier FROM va)

# 6.7. Aggregate functions

Aggregate functions are impemented. *Note that calculations as SUM(balance)+2 are not allowed.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **EXPR-1** | **EXPR-2** | **EXPR-3** | **EXPR-4** | **EXPR-5** |
| 1 | 9 | 48700 | 25000 | 0 | 5411.111111 |

13. Aggregate functions COUNT, SUM, MAX, MIN, AVG

SELECT COUNT(\*),SUM(balance),MAX(balance),MIN(balance),AVG(balance)

FROM le

More tables may be joined.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **No** | **Name** | **Name** |
| 1 | 100 | HUMBER LTD. | HUMBER LTD. |
| 2 | 102 | AX & AX LTD. | AX & AX LTD. |
| 3 | 105 | WEBB'S SUPPLIERS LTD. | WEBB'S SUPPLIERS LTD. |
| 4 | 111 | TRAWSOM LTD. | TRAWSOM LTD. |
| 5 | 123 | BRAUN GMBH | BRAUN GMBH |
| 6 | 205 | SCHIERMACHER LTD. | SCHIERMACHER LTD. |
| 7 | 260 | CLORID LTD. | CLORID LTD. |
| 8 | 270 | OHIO INC. | OHIO INC. |
| 9 | 271 | DANDY INC. | DANDY INC. |

14. Using the same table multiple times

SELECT no,name, b.name

FROM le a, le b

WHERE a.no=b.no

# 6.8. LIKE and MATCHES

Like may be used for search on sting patterns as **"a\_b%c[^a-kp]"** The ODBC like escape clause is supported

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 2 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 3 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |
| 4 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 5 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |

15. Using the LIKE function

SELECT \*

FROM va

WHERE name NOT LIKE "%O%" { escape 'x' }

Matches offers another search method with patterns as **"a?b\*c[^a-kp]"**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 2 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 3 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |

16. Using the MATCHES function

SELECT \*

FROM va

WHERE name MATCHES "\*CH\*"

# 6.9. How to use functions within SELECT statements

Functions may be called directly of by use of the { fn ... } clause.

|  |  |  |
| --- | --- | --- |
|  | **No** | **EXPR-1** |
| 1 | 0101 | chocolate |
| 2 | 0102 | large machine |
| 3 | 0110 | bus |
| 4 | 1001 | money |
| 5 | 1005 | machine |
| 6 | 2001 | creditcard |
| 7 | 2002 | id-card |

17. Calling functions within SELECT

SELECT no,{ fn LCASE(name) }

FROM va

The full ODBC syntax for functions calls may also be used.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **No** | **EXPR-1** | **Name** |
| 1 | 0101 | 101 | CHOCOLATE |
| 2 | 0102 | 102 | LARGE MACHINE |
| 3 | 0110 | 110 | BUS |
| 4 | 1001 | 1001 | MONEY |
| 5 | 1005 | 1005 | MACHINE |
| 6 | 2001 | 2001 | CREDITCARD |
| 7 | 2002 | 2002 | ID-CARD |

18. Calling functions within SELECT with full ODBC syntax

SELECT no,--(\*vendor(SWTools),product(ODBC) fn CONVERT(no,SQL\_INTEGER)\*)--,name

FROM va

# 6.10. Date, Time and timestamp

Date, Time and Timestamp values may be stated by { d 'yyyy-mm-dd' }, { t 'hh:mm:ss' } and { ts 'yyyy-mm-dd hh:mm:ss' }

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 2 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |
| 3 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 4 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |

19. Using Date, Time and Timestamp syntax

SELECT \*

FROM articles

WHERE 'last purchase' > { d '1994-06-01' }

Dates stored in the files as YYMMDD or DDMMYY will be turned to correct SQLDate YYYY-MM-DD when the format is given as ,6, or ,8,

Timestamp data are asumed to be stored as 14 digits numeric YYYYMMDDHHMMSS, fractions of seconds are not supported.

The standard SQL syntax may also be used:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 2 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |
| 3 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 4 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |

20. Using standard SQL syntax for dates

SELECT \*

FROM articles

WHERE 'last purchase' > #1994-06-01#

# 6.11. Using field numbers

As an extension to SQL fieldnumbers may be given instead of fieldnames.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Suppli** | **Price** | **No** | **Name** |
| 1 | 0110 | BUS | 123 | 100000 | 123 | BRAUN GMBH |
| 2 | 0102 | LARGE MACHINE | 100 | 20000 | 100 | HUMBER LTD. |
| 3 | 1005 | MACHINE | 100 | 2000 | 100 | HUMBER LTD. |
| 4 | 1001 | MONEY | 205 | 1000 | 205 | SCHIERMACHER LTD. |

21. Using field numbers instead of field names

SELECT #1-2,6,price,a.#1-2

FROM va,le a

WHERE #3>100 AND le#1=#6

ORDER BY #2

Recordnumbers may be referred with RECNO, Relative recordnumber with NUMBER.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **OrderID** | **RECNO** | **NUMBER** | **ProdID** | **NUMBER** | **RECNO** |
| 1 | 34 | 801 | 22 | 31 | 22 | 38 |
| 2 | 34 | 802 | 23 | 32 | 23 | 39 |
| 3 | 52 | 803 | 24 | 33 | 24 | 40 |
| 4 | 52 | 804 | 25 | 34 | 25 | 41 |
| 5 | 52 | 805 | 26 | 35 | 26 | 42 |
| 6 | 52 | 806 | 27 | 36 | 27 | 43 |
| 7 | 48 | 807 | 28 | 39 | 28 | 44 |
| 8 | 48 | 808 | 29 | 40 | 29 | 45 |
| 9 | 48 | 809 | 30 | 41 | 30 | 46 |

SELECT OrderID,recno,number,a.ProdID,a.number,a.recno FROM Orders,Product a

WHERE a.NUMBER=NUMBER and recno<810 and recno>800

Any calculations may be given, including operations on TABLE (subscripted) fields

# 6.12. Field subscriptions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **EXPR-1** | **Cost** |
| 1 | 0101 | CHOCOLATE | 2 | 1.5 | 1.5 |
| 2 | 0102 | LARGE MACHINE | 20000 | 10000 | 10000 |
| 3 | 0110 | BUS | 100000 | 60000 | 60000 |
| 4 | 1001 | MONEY | 1000 | 500 | 500 |
| 5 | 1005 | MACHINE | 2000 | 1500 | 1500 |
| 6 | 2001 | CREDITCARD | 25 | 10 | 10 |
| 7 | 2002 | ID-CARD | 25 | 10 | 10 |

22. Subscribed fields

SELECT no, name, price, price(1), cost

FROM va

# 6.13. GROUP BY, HAVING, DISTINCT and UNION

The GROUP BY may be used to form groups of aggregate functions

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Suppli** | **EXPR-1** | **EXPR-2** |
| 1 | 100 | 2 | 22000 |
| 2 | 123 | 1 | 100000 |
| 3 | 205 | 3 | 1050 |
| 4 | 271 | 1 | 2 |

23. A simple GROUP BY sample

SELECT supplier,COUNT(\*),SUM(price)

FROM va

GROUP BY supplier

Having is a selection after the grouping has been done

|  |  |  |
| --- | --- | --- |
|  | **Suppli** | **EXPR-1** |
| 1 | 123 | 100000 |
| 2 | 271 | 2 |

24. A simple HAVING sample

SELECT supplier,SUM(price)

FROM va

GROUP BY supplier

HAVING COUNT(\*)=1

By use of DISTINCT all columns with the same contents are suppressed

|  |  |  |
| --- | --- | --- |
|  | **Suppli** | **Group** |
| 1 | 271 | 0 |
| 2 | 100 | 9 |
| 3 | 123 | 2 |
| 4 | 205 | 0 |
| 5 | 100 | 1 |
| 6 | 205 | 9 |

25. SELECT using DISTINCT

SELECT DISTINCT supplier,group

FROM va

The DISTINCT may also suppress values when used with the aggregate functions

|  |  |  |
| --- | --- | --- |
|  | **EXPR-1** | **EXPR-2** |
| 1 | 123027 | 4 |

26. SELECT using DISTINCT on aggregate functions

SELECT SUM(DISTINCT price), COUNT(DISTINCT supplier)

FROM va

UNIONs of select statements may be formed, UNION ALL is supported.

|  |  |  |
| --- | --- | --- |
|  | **No** | **Cost** |
| 1 | 0101 | 1.5 |
| 2 | 2002 | 10 |
| 3 | 2001 | 10 |
| 4 | 1001 | 500 |
| 5 | 1001 | 1000 |
| 6 | 1005 | 2000 |
| 7 | 0102 | 20000 |
| 8 | 0110 | 100000 |

27. SELECT using UNIONs

SELECT no,price

FROM va

WHERE price>100 UNION ALL

SELECT no,cost FROM va WHERE cost<1000 ORDER BY 2

Anywhere a SELECT statement can be used, the VALUES table constructor may be used.

# 6.14. VALUES constructor and SELECT from result set

|  |  |  |  |
| --- | --- | --- | --- |
|  | **V1** | **V2** | **V3** |
| 1 | 4701 | aa | 65 |
| 2 | 4702 | bb | 8 |

28. VALUES constructor

SELECT \*

FROM VALUES ("4701","aa",65),("4702","bb",8)

SELECT from a resultset is also possible.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Suppli** | **Price** |
| 1 | 0101 | CHOCOLATE | 271 | 2 |
| 2 | 0102 | LARGE MACHINE | 100 | 20000 |
| 3 | 0110 | BUS | 123 | 100000 |
| 4 | 1001 | MONEY | 205 | 1000 |
| 5 | 1005 | MACHINE | 100 | 2000 |
| 6 | 2001 | CREDITCARD | 205 | 25 |
| 7 | 2002 | ID-CARD | 205 | 25 |

29. SELECT from result set

SELECT \*

FROM (SELECT no,name,supplier,price FROM VA)

Note by joining tables the WHERE becomes really importent. If no where is stated, the joined table is read once for each element in the first table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Curre** | **Name** | **Rate** | **Curre** | **Name** | **Rate** |
| 1 | 0 | UKP | 100 | 0 | UKP | 100 |
| 2 | 0 | UKP | 100 | 1 | DEM | 380.59 |
| 3 | 0 | UKP | 100 | 2 | USD | 626.65 |
| 4 | 1 | DEM | 380.59 | 0 | UKP | 100 |
| 5 | 1 | DEM | 380.59 | 1 | DEM | 380.59 |
| 6 | 1 | DEM | 380.59 | 2 | USD | 626.65 |
| 7 | 2 | USD | 626.65 | 0 | UKP | 100 |
| 8 | 2 | USD | 626.65 | 1 | DEM | 380.59 |
| 9 | 2 | USD | 626.65 | 2 | USD | 626.65 |

30. Joined tables without where

SELECT \*,a.\*

FROM ku,ku a

# 7. Updating the database and datadictionary itself

The SWODBC driver may be delivered for read-only or optionally with update for file interfaces allowing this.

The INTO TEMP clause creates a file and filedefinition with the given name. This file exists until you manually deletes it with DROP TABLE which makes INTO TEMP an easy way to export a file to another filesystem.

The filename may be qualified by: xx\yyyy.name, where

xx = Desided file ID, if omitted or invalid the driver selects a free ID

yyyy = BASIS file interface name (owner), defaults to the first (SSV).

The file definition will be marked TEMP, any existing TEMP file is overwritten. The ORDER BY (or GROUP BY) is used to define the file index.If omitted an index as #1,NP is used.

# 7.1. Copying table INTO TEMP

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 2 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 3 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 4 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |
| 5 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 6 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |
| 7 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |

SELECT \*

FROM va

ORDER BY 2

INTO TEMP mytable

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 2 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 3 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 4 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |
| 5 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 6 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |
| 7 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |

SELECT \*

FROM mytable

# 7.2. INSERT values INTO table

By use of INSERT...VALUES new records can be created

*Query executed - No results returned.*

INSERT INTO mytable

VALUES (1,2,3)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 1 | 2 | 3 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |
| 2 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 3 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 4 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 5 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |
| 6 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 7 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |
| 8 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |

SELECT \*

FROM mytable

# 7.3. INSERT values from other tables

Records from other tables can be copied with INSERT...SELECT

*Query executed - No results returned.*

INSERT INTO mytable

(SELECT no,name,balance FROM le WHERE balance>10000)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 1 | 2 | 3 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |
| 2 | 102 | AX & AX LTD. | 25000 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |
| 3 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 4 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 5 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 6 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |
| 7 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 8 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |
| 9 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |
| 10 | 205 | SCHIERMACHER LTD. | 20000 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |

SELECT \*

FROM mytable

Insert columns may be given and the value table contructor may be used to form multiple records

*Query executed - No results returned.*

INSERT INTO mytable

(no,name,price) VALUES ("4701","aa",65),("4702","bb",8)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 1 | 2 | 3 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |
| 2 | 102 | AX & AX LTD. | 25000 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |
| 3 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 4 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 5 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 6 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |
| 7 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 8 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |
| 9 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |
| 10 | 205 | SCHIERMACHER LTD. | 20000 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |
| 11 | 4701 | aa | 65 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |
| 12 | 4702 | bb | 8 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |

SELECT \*

FROM mytable

Together with the select specific columns can be moved

*Query executed - No results returned.*

INSERT INTO mytable

(no,name,holding)

(SELECT no,name,balance FROM le WHERE balance>0 AND balance<10000)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 1 | 2 | 3 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |
| 2 | 102 | AX & AX LTD. | 25000 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |
| 3 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 4 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 5 | 260 | CLORID LTD. | 0 | 0 | 0000-00-00 |  | 0 | 2000 | 0 | 0 |
| 6 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 7 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |
| 8 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 9 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |
| 10 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |
| 11 | 270 | OHIO INC. | 0 | 0 | 0000-00-00 |  | 0 | 200 | 0 | 0 |
| 12 | 205 | SCHIERMACHER LTD. | 20000 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |
| 13 | 111 | TRAWSOM LTD. | 0 | 0 | 0000-00-00 |  | 0 | 1000 | 0 | 0 |
| 14 | 105 | WEBB'S SUPPLIERS LTD | 0 | 0 | 0000-00-00 |  | 0 | 500 | 0 | 0 |
| 15 | 4701 | aa | 65 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |
| 16 | 4702 | bb | 8 | 0 | 0000-00-00 |  | 0 | 0 | 0 | 0 |

SELECT \*

FROM mytable

# 7.4. Updating existing records

Existing records can be updated with the UPDATE searched statement

*Query executed - No results returned.*

UPDATE mytable

SET holding=price+100, 'last purchase'=19960331

WHERE cost=0

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 1 | 2 | 3 | 0 | 1996-03-31 |  | 0 | 103 | 0 | 0 |
| 2 | 102 | AX & AX LTD. | 25000 | 0 | 1996-03-31 |  | 0 | 25100 | 0 | 0 |
| 3 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 4 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 5 | 260 | CLORID LTD. | 0 | 0 | 1996-03-31 |  | 0 | 100 | 0 | 0 |
| 6 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 7 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |
| 8 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 9 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |
| 10 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |
| 11 | 270 | OHIO INC. | 0 | 0 | 1996-03-31 |  | 0 | 100 | 0 | 0 |
| 12 | 205 | SCHIERMACHER LTD. | 20000 | 0 | 1996-03-31 |  | 0 | 20100 | 0 | 0 |
| 13 | 111 | TRAWSOM LTD. | 0 | 0 | 1996-03-31 |  | 0 | 100 | 0 | 0 |
| 14 | 105 | WEBB'S SUPPLIERS LTD | 0 | 0 | 1996-03-31 |  | 0 | 100 | 0 | 0 |
| 15 | 4701 | aa | 65 | 0 | 1996-03-31 |  | 0 | 165 | 0 | 0 |
| 16 | 4702 | bb | 8 | 0 | 1996-03-31 |  | 0 | 108 | 0 | 0 |

SELECT \*

FROM mytable

# 7.5. DELETE multiple records

The searched DELETE removes one or several records

*Query executed - No results returned.*

DELETE FROM mytable

WHERE price<100 AND cost=0

After the delete the function SQLRowCount delivers number of rows updated:

|  |  |
| --- | --- |
|  | **Rows** |
| 1 | 7 |

SQLRowCount(hstmt)

And the resulting table looks like:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 102 | AX & AX LTD. | 25000 | 0 | 1996-03-31 |  | 0 | 25100 | 0 | 0 |
| 2 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 3 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 4 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 5 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |
| 6 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 7 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |
| 8 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |
| 9 | 205 | SCHIERMACHER LTD. | 20000 | 0 | 1996-03-31 |  | 0 | 20100 | 0 | 0 |

SELECT \*

FROM mytable

# 7.6. GRANT/REVOKE implimentation

The GRANT and REVOKE statements are implemented as just dummies as user priviledges are not maintained in the data dictionary.

*Query executed - No results returned.*

GRANT SELECT ON mytable TO somebody

*Query executed - No results returned.*

REVOKE SELECT ON mytable FROM somebody

# 7.7. DROP table

Using DROP TABLE a table and its definition can be removed.

*Query executed - No results returned.*

DROP TABLE mytable

# 8. Current of cursors

To avoid changes in the demo system we duplicate

- va

- le

into

- SWva

- SWle

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 2 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 3 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 4 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |
| 5 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |
| 6 | 2001 | CREDITCARD | 25 | 10 | 1995-01-01 | 205 | 9 | 10 | 102 | 0 |
| 7 | 2002 | ID-CARD | 25 | 10 | 1994-06-30 | 205 | 9 | 200 | 102 | 0 |

SELECT \*

FROM va

ORDER BY 1

INTO TEMP zb\SWva

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Address** | **Town** | **Curre** | **Balance** |
| 1 | 100 | HUMBER LTD. | HUMBER STREET 223 | 4711 COPENHAGEN S | 0 | 0 |
| 2 | 102 | AX & AX LTD. | SEA PARK ROAD 43 | 2100 COPENHAGEN | 0 | 25000 |
| 3 | 105 | WEBB'S SUPPLIERS LTD. | EAST STREET 373 | 4711 COPENHAGEN F | 0 | 500 |
| 4 | 111 | TRAWSOM LTD. | WEST STREET 111 | 1820 COPENHAGEN C | 0 | 1000 |
| 5 | 123 | BRAUN GMBH | PLATZ DAMN 12 | LUXEMBOURG | 1 | 0 |
| 6 | 205 | SCHIERMACHER LTD. | BOULEVARD ROYAL 63 | LUXEMBOURG | 1 | 20000 |
| 7 | 260 | CLORID LTD. | COPENHAGEN STREET 3 | 1154 COPENHAGEN K | 0 | 2000 |
| 8 | 270 | OHIO INC. | MAIN AVENUE | NEW YORK | 2 | 200 |
| 9 | 271 | DANDY INC. | 13-MAIN STREET | LOS ANGELES | 2 | 0 |

SELECT \*

FROM le

ORDER BY 1

INTO TEMP zc\SWle

# 8.1. Getting CURSOR name

Cursors are named, the name can be retrieved by SQLGetCursorName:

|  |  |
| --- | --- |
|  | **Cursorname** |
| 1 | SQL\_CUR00001 |

SQLGetCursorName(hstmt,cursorname,256,&len)

And the cursor name can be set by SQLSetCursorName before the SELECT is done.

# 8.2. Setting CURSOR name

*Query executed - No results returned.*

SQLSetCursorName(hstmt,"mycursor",SQL\_NTS)

# 8.3. SELECT for UPDATE

The select for UPDATE is implemented. Note both files may be updated.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Group** | **Sup** | **Sname** |
| 1 | 0101 | CHOCOLATE | 2 | 1.5 | 0 | 271 | DANDY INC. |
| 2 | 0102 | LARGE MACHINE | 20000 | 10000 | 9 | 100 | HUMBER LTD. |
| 3 | 0110 | BUS | 100000 | 60000 | 2 | 123 | BRAUN GMBH |
| 4 | 1001 | MONEY | 1000 | 500 | 0 | 205 | SCHIERMACHER LTD. |
| 5 | 1005 | MACHINE | 2000 | 1500 | 1 | 100 | HUMBER LTD. |
| 6 | 2001 | CREDITCARD | 25 | 10 | 9 | 205 | SCHIERMACHER LTD. |
| 7 | 2002 | ID-CARD | 25 | 10 | 9 | 205 | SCHIERMACHER LTD. |

SELECT no,name,price,cost,group,a.no Sup,a.name Sname

FROM SWva,OUTER SWle a

WHERE a.no=supplier

FOR UPDATE OF no,price,supplier,balance

As an extension to the SQL for these examples, the cursor can be positioned using **SELECT ... WHERE CURRENT OF cursorname = rownumber**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 2001 | CREDITCARD | 25 | 10 | 0000-00-00 |  | 9 | 0 | 0 | 0 |

SELECT \*

FROM SWva

WHERE CURRENT OF mycursor=6

*Query executed - No results returned.*

UPDATE SWva

SET price=price\*1.25,group=2

WHERE CURRENT OF mycursor

# 8.4. SELECT from cursor

The result may be retrieved again by the extension select

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 2001 | CREDITCARD | 31.25 | 10 | 0000-00-00 |  | 2 | 0 | 0 | 0 |

SELECT \*

FROM SWva

WHERE CURRENT OF mycursor

Also rows from joined tables may be used in positioned update

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Address** | **Town** | **Curre** | **Balance** |
| 1 | 123 | BRAUN GMBH |  |  | 0 | 0 |

SELECT \*

FROM SWle

WHERE CURRENT OF mycursor = 3

*Query executed - No results returned.*

UPDATE SWle

SET balance = 4711

WHERE CURRENT OF mycursor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Address** | **Town** | **Curre** | **Balance** |
| 1 | 102 | AX & AX LTD. | SEA PARK ROAD 43 | 2100 COPENHAGEN | 0 | 25000 |
| 2 | 105 | WEBB'S SUPPLIERS LTD. | EAST STREET 373 | 4711 COPENHAGEN F | 0 | 500 |
| 3 | 111 | TRAWSOM LTD. | WEST STREET 111 | 1820 COPENHAGEN C | 0 | 1000 |
| 4 | 123 | BRAUN GMBH | PLATZ DAMN 12 | LUXEMBOURG | 1 | 4711 |
| 5 | 205 | SCHIERMACHER LTD. | BOULEVARD ROYAL 63 | LUXEMBOURG | 1 | 20000 |
| 6 | 260 | CLORID LTD. | COPENHAGEN STREET 3 | 1154 COPENHAGEN K | 0 | 2000 |
| 7 | 270 | OHIO INC. | MAIN AVENUE | NEW YORK | 2 | 200 |

SELECT \*

FROM SWle

WHERE balance > 0

Update of a row can be performed twice:

*Query executed - No results returned.*

UPDATE SWle

SET balance=0

WHERE CURRENT OF mycursor

# 8.5. DELETE from cursor

The positioned delete can be done:

*Query executed - No results returned.*

DELETE FROM SWva

WHERE CURRENT OF mycursor=6

The FOR UPDATE may be given without fields if only DELETE should follow

|  |  |  |  |
| --- | --- | --- | --- |
|  | **No** | **Suppli** | **Cost** |
| 1 | 0101 | 271 | 1.5 |
| 2 | 0102 | 100 | 10000 |
| 3 | 0110 | 123 | 60000 |
| 4 | 1001 | 205 | 500 |
| 5 | 1005 | 100 | 1500 |
| 6 | 2002 | 205 | 10 |

SELECT no,supplier,cost

FROM SWva FOR UPDATE

*Query executed - No results returned.*

DELETE FROM SWva

WHERE CURRENT OF mycursor

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **No** | **Name** | **Price** | **Cost** | **Last purchase** | **Suppli** | **Group** | **Holding** | **Altern** | **Free** |
| 1 | 0101 | CHOCOLATE | 2 | 1.5 | 1995-01-01 | 271 | 0 | 100 | 270 | 0 |
| 2 | 0102 | LARGE MACHINE | 20000 | 10000 | 1993-01-01 | 100 | 9 | 0 | 0 | 0 |
| 3 | 0110 | BUS | 100000 | 60000 | 1993-12-15 | 123 | 2 | 1 | 100 | 0 |
| 4 | 1001 | MONEY | 1000 | 500 | 1994-12-31 | 205 | 0 | 100 | 102 | 0 |
| 5 | 1005 | MACHINE | 2000 | 1500 | 1994-06-01 | 100 | 1 | 10 | 0 | 0 |

SELECT \*

FROM SWva

# 9. Views

A view may be created defining a select

*Query executed - No results returned.*

CREATE VIEW myview (A,B,C)

AS (SELECT no, name, holding FROM va WHERE holding > 0)

Selecting fields from a view first executes the defined select. The table definition but not the table itself exists. A view cannot be updated.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **A** | **B** | **C** |
| 1 | 0110 | BUS | 1 |
| 2 | 1005 | MACHINE | 10 |
| 3 | 2001 | CREDITCARD | 10 |
| 4 | 2002 | ID-CARD | 200 |

SELECT \*

FROM myview

WHERE C <> 100

The view may be removed afterwards:

*Query executed - No results returned.*

DROP VIEW myview

# 10. Create / Alter and Rename tables

The CREATE/ALTER TABLE has the following extensions to the standard SQL:

a. Table name can be given as described for SELECT...INTO

b. Field formats may be given including Pack options/Bytes etc.

c. PRIMARY KEY may specify SWTools key syntax using fieldnumbers

# 10.1. How to create tables

*Query executed - No results returned.*

CREATE TABLE mytable (no SHORT(4),

name CHAR(20),

balance NUMERIC(8,2),

PRIMARY KEY (name ASC,no DESC))

*Query executed - No results returned.*

CREATE TABLE 'yourtable' ('no a' SHORT ( 4 ) UNIQUE,

'name b' CHAR ( 20 ) ,

balance DECIMAL ( 8 , 2 ) )

*Query executed - No results returned.*

CREATE TABLE sometable (no LONG ,

name CHAR(20),

PRIMARY KEY(#1,#2,NP))

Index can be created and dropped again

*Query executed - No results returned.*

CREATE UNIQUE INDEX abcdef ON mytable (no ASC,name DESC)

*Query executed - No results returned.*

DROP INDEX mytable.abcdef

# 10.2. ALTER table definition

The ALTER TABLE supports ADD,DROP and MODIFY of columns

*Query executed - No results returned.*

ALTER TABLE mytable ADD (date NUMERIC(,8,2P7),code CHAR(13)),

DROP COLUMN balance,name,

MODIFY no NUMERIC

A table may be renamed

*Query executed - No results returned.*

RENAME TABLE mytable TO agoodtable

# 11. Data types

The below mentioned data types returned by SQLGetTypeInfo are valid. The use of NULL values are resticted due to the file systems involved.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **TYPE\_NAME** | **DATA** | **PRECI** | **LPRE** | **LSUF** | **CREATE\_PARAMS** | **NUL** | **CAS** | **SEA** | **UNS** | **MO** | **AUTO** | **LOC** | **MIN** | **MAX** |
| 1 | CHAR | 1 | 254 | " | " | MAX LENGTH | 0 | 1 | 3 | NUL | 0 | NULL | NUL | NUL | NUL |
| 2 | NUMERIC | 8 | 15 | NULL | NULL | PRECISION,SCALE | 0 | 0 | 2 | 0 | 0 | 0 | NUL | 0 | 9 |
| 3 | DECIMAL | 8 | 15 | NULL | NULL | PRECISION,SCALE | 0 | 0 | 2 | 0 | 0 | 0 | NUL | 0 | 9 |
| 4 | LONG | 4 | 10 | NULL | NULL | PRECISION | 0 | 0 | 2 | 0 | 0 | 0 | NUL | 0 | 0 |
| 5 | SHORT | 5 | 5 | NULL | NULL | PRECISION | 0 | 0 | 2 | 0 | 0 | 0 | NUL | 0 | 0 |
| 6 | FLOAT | 8 | 15 | NULL | NULL | PRECISION,SCALE | 0 | 0 | 2 | 0 | 0 | 0 | NUL | 0 | 9 |
| 7 | REAL | 8 | 15 | NULL | NULL | PRECISION,SCALE | 0 | 0 | 2 | 0 | 0 | 0 | NUL | 0 | 9 |
| 8 | DOUBLE | 8 | 15 | NULL | NULL | PRECISION,SCALE | 0 | 0 | 2 | 0 | 0 | 0 | NUL | 0 | 9 |
| 9 | DATE | 9 | 10 | # | # | NULL | 1 | 0 | 2 | NUL | 0 | NULL | NUL | NUL | NUL |
| 10 | TIME | 10 | 8 | # | # | NULL | 1 | 0 | 2 | NUL | 0 | NULL | NUL | NUL | NUL |
| 11 | TIMESTAMP | 11 | 19 | # | # | NULL | 1 | 0 | 2 | NUL | 0 | NULL | NUL | NUL | NUL |
| 12 | VARCHAR | 12 | 1024 | " | " | MAX LENGTH | 0 | 1 | 3 | NUL | 0 | NULL | NUL | NUL | NUL |

SQLGetTypeInfo(hstmt,SQL\_ALL\_TYPES)

Fieldnames are taken from the Data-Dictionary SQLnames, if these are not present the normal fieldname is used, ' \ . and " will be replaced by space. In case of duplicate fieldnames 1 is added to the last character in the name. Fields without name or format definitions is omitted.

A description for the **'Payment Terms'** table comes like:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **QUALI** | **OWNER** | **TABLE\_NAME** | **COLUMN\_NAME** | **TYPE** | **TYPE\_NAME** | **PREC** | **LEN** | **SCALE** | **RADIX** | **NULL** | **REMARKS** |
| 1 | NULL | NULL | Orders | CustID | 12 | VARCHAR | 5 | 5 | 0 | NULL | 0 | NULL |
| 2 | NULL | NULL | Orders | OrderDate | 9 | DATE | 10 | 6 | 0 | 10 | 1 | NULL |
| 3 | NULL | NULL | Orders | OrderID | 4 | INTEGER | 5 | 4 | 0 | 10 | 0 | NULL |
| 4 | NULL | NULL | Orders | ProdID | 4 | INTEGER | 5 | 4 | 0 | 10 | 0 | NULL |
| 5 | NULL | NULL | Orders | Quantity | 4 | INTEGER | 5 | 4 | 0 | 10 | 0 | NULL |

SQLColumns(hstmt,NULL,0,NULL,0,"Orders",SQL\_NTS,NULL,0)

After a SELECT SQLDescribeCol may look like the following:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **OrderID** | **CustID** | **ProdID** | **OrderDate** | **Quantity** |
| 1 | 47 | SEVES | 52 | 1990-11-15 | 20 |

SELECT \*

FROM Orders

WHERE OrderID = 47

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Name** | **SQL-Type** | **Precision** | **Scale** | **Nullable** |
| 1 | OrderID | 4 SQL\_INTEGER | 5 | 0 | 0 |
| 2 | CustID | 12 SQL\_VARCHAR | 5 | 0 | 0 |
| 3 | ProdID | 4 SQL\_INTEGER | 5 | 0 | 0 |
| 4 | OrderDate | 9 SQL\_DATE | 10 | 0 | 1 |
| 5 | Quantity | 4 SQL\_INTEGER | 5 | 0 | 0 |

SQLDescribeCol(hstmt,\*,name,256,&len,&type,&precision,&scale,&nullable)

And the more detailed column attributes:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Aut** | **Cas** | **Cou** | **Siz** | **Label** | **Len** | **M** | **Name** | **Nul** | **Own** | **Prec** | **Qua** | **Sca** | **Sea** | **Tab** | **Typ** | **Typname** | **Uns** | **Updat** |
| 1 | 0 | 0 | 5 | 5 | Order | 4 | 0 | Order | 0 |  | 5 |  | 0 | 2 |  | 4 | INTEGER | 1 | 1 |
| 2 | 0 | 1 | 5 | 5 | CustI | 5 | 0 | CustI | 0 |  | 5 |  | 0 | 3 |  | 12 | VARCHAR | 1 | 1 |
| 3 | 0 | 0 | 5 | 5 | ProdI | 4 | 0 | ProdI | 0 |  | 5 |  | 0 | 2 |  | 4 | INTEGER | 1 | 1 |
| 4 | 0 | 0 | 5 | 10 | Order | 6 | 0 | Order | 1 |  | 10 |  | 0 | 2 |  | 9 | DATE | 1 | 1 |
| 5 | 0 | 0 | 5 | 6 | Quant | 4 | 0 | Quant | 0 |  | 5 |  | 0 | 2 |  | 4 | INTEGER | 0 | 1 |

SQLColAttributes(hstmt,\*,\*,info,256,&len,&val)

The SQLSpecialColumns gives the best access key to the table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **SCOPE** | **COLUMN\_NAME** | **DATA\_TYPE** | **TYPE\_NAME** | **PREC** | **LEN** | **SCALE** | **PSEUDO** |
| 1 | 1 | CustID | 0 | 12 | 0 | 5 | 5 | 1 |
| 2 | 1 | OrderID | 0 | 4 | 0 | 5 | 4 | 1 |
| 3 | 1 | ProdID | 0 | 4 | 0 | 5 | 4 | 1 |

SQLSpecialColumns(hstmt,SQL\_BEST\_ROWID,

NULL,0,NULL,0,"Orders",SQL\_NTS,

SQL\_SCOPE\_CURROW,SQL\_NULLABLE)

Whereas SQLStatistics provides information of the table and the single keyparts

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **QUALI** | **OWNER** | **TABLE\_NAME** | **UNI** | **XQUALI** | **INDEX\_NAME** | **TYP** | **SEQ** | **COLUMN\_NAME** | **COL** | **CAR** | **PAGES** | **FIL** |
| 1 | NULL | NULL | Orders | NUL | Orders | NULL | 0 | NUL | NULL | NUL | 300 | 51 | NUL |
| 2 | NULL | NULL | Orders | 0 | Orders | INDEX01 | 3 | 1 | OrderID | A | 300 | 37 | NUL |
| 3 | NULL | NULL | Orders | 0 | Orders | INDEX01 | 3 | 2 | CustID | A | 300 | 37 | NUL |
| 4 | NULL | NULL | Orders | 0 | Orders | INDEX01 | 3 | 3 | ProdID | A | 300 | 37 | NUL |
| 5 | NULL | NULL | Orders | 0 | Orders | INDEX02 | 3 | 1 | CustID | A | 300 | 37 | NUL |
| 6 | NULL | NULL | Orders | 0 | Orders | INDEX02 | 3 | 2 | OrderID | A | 300 | 37 | NUL |
| 7 | NULL | NULL | Orders | 0 | Orders | INDEX02 | 3 | 3 | ProdID | A | 300 | 37 | NUL |
| 8 | NULL | NULL | Orders | 1 | Orders | INDEX03 | 3 | 1 | ProdID | A | 300 | 37 | NUL |
| 9 | NULL | NULL | Orders | 1 | Orders | INDEX03 | 3 | 2 | OrderID | A | 300 | 37 | NUL |

SQLStatistics(hstmt,NULL,0,NULL,0,"Orders",SQL\_NTS,SQL\_INDEX\_ALL,SQL\_ENSURE)

Note: SQL\_ENSURE is required to get the correct values of Cardinality and pages. For TABLE\_STAT Cardinality is total number of records, Pages the files size in KB. For INDEX Cardinality is also total number of records, Pages the index size in KB.

# 12. Table types, names, Owners and Qualifiers

The table names is decided from the FNAME= and the FNAMELEN= parameters stated in ODBC.INI for the data source or given in the connection string to SQLDriverConnect. \* marks the default.

FNAME=n How to use table names

0 File ID is always used

1 \* If SID is filled, use the first 11 characters of this else same as

2 Use reduced FILENAME according to following rules:

a. Start from first alpha character in the name

b. If spaces is present, start after the last space found

c. If : \ or / is present, start after the last of these

d. If name becomes XX.xxx, remove XX.

e. If name ends with abc, remove abc.

3 Use FILETEXT as tablename until first non-alphanumeric character.

4 Use FILETEXT as tablename

FNAMELEN=n Length of Table name

0 No restrictions on tablename

1 \* Tablename is delimited by the first occurence of a space

>2 Tablename will be of maximum this size.

The characters \ . ' and " in any file- or fieldname will be replaced by space as not all database programs is able to handle these.

If the tablename becomes invalid or if a duplicate name is found the ID is used.

The table informations also uses the following:

OWNER=n Usage of owners

0 No owners, NULL is returned

1 \* Use file typename as owner

2 Use file ID as owner

QUALIFIER=n Usage of file qualifiers

0 No qualifiers, NULL is returned

1\* Use file ID as qualifier

2 Use file typename as qualifier

FTEXT=n Usage of file text description

0 \* The file text is used

1 Filename

2 Filename + File text

3 File ID + Filename + File text

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **TABLE\_QUALIFIER** | **TABLE\_OWNER** | **TABLE\_NAME** | **TABLE\_TYPE** | **REMARKS** |
| 1 | NULL | NULL | SY | SYSTEM TABLE | Systemfields |
| 2 | NULL | NULL | AF-0500000 | TABLE | Sales order 00/header rec |
| 3 | NULL | NULL | AF-0500020 | TABLE | Sales order 20/ Item reco |
| 4 | NULL | NULL | Articles | TABLE | Article file |
| 5 | NULL | NULL | Currency | TABLE | Currency file |
| 6 | NULL | NULL | Customer | TABLE | ODBC Customer |
| 7 | NULL | NULL | GF-03000 | TABLE | Debtor/creditor master 1 |
| 8 | NULL | NULL | GF-03100 | TABLE | Debtor/creditor transacti |
| 9 | NULL | NULL | Groups | TABLE | Article groups |
| 10 | NULL | NULL | LF-06000 | TABLE | Article master |
| 11 | NULL | NULL | LF-060011 | TABLE | Stock location 1 |
| 12 | NULL | NULL | Orders | TABLE | ODBC Orders |
| 13 | NULL | NULL | Product | TABLE | ODBC Product |
| 14 | NULL | NULL | Suppliers | TABLE | Supplier file |

SQLTables(hstmt,NULL,0,NULL,0,NULL,0,NULL,0)

This may be modified in ODBC.INI or with the connection parameters:

|  |  |
| --- | --- |
|  | **Resulting connection string** |
| 1 | DSN=SWTools32,Fname=3,Owner=0,Qualifier=0,Ftext=3,,COM=001,Description=SW-Tools 32 Bit ODB |

SQLDriverConnect(henv,NULL,

"DSN=SWTools32;Fname=3;Owner=0;Qualifier=0;Ftext=3",

SQL\_NTS,constr,256,&len,SQL\_DRIVER\_COMPLETE)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **TABLE\_QUALIFIER** | **TABLE\_OWNER** | **TABLE\_NAME** | **TABLE\_TYPE** | **REMARKS** |
| 1 | NULL | NULL | Systemfields | SYSTEM TABLE | SY Systemfields |
| 2 | NULL | NULL | AU | TABLE | AU AF-05000abc Sales order 20/ I |
| 3 | NULL | NULL | Article | TABLE | D4 LF-06000abc Article master |
| 4 | NULL | NULL | Currency | TABLE | KU Currency file |
| 5 | NULL | NULL | Debtor | TABLE | JH GF-03000abc Debtor/creditor m |
| 6 | NULL | NULL | GR | TABLE | GR Article groups |
| 7 | NULL | NULL | JI | TABLE | JI GF-03100abc Debtor/creditor t |
| 8 | NULL | NULL | ODBC | TABLE | OC 9/customer ODBC Customer |
| 9 | NULL | NULL | OP | TABLE | OP 9/product ODBC Product |
| 10 | NULL | NULL | OR | TABLE | OR 9/orders ODBC Orders |
| 11 | NULL | NULL | Sales | TABLE | AS AF-05000abc Sales order 00/he |
| 12 | NULL | NULL | Stock | TABLE | D7 LF-06001abc Stock location 1 |
| 13 | NULL | NULL | Supplier | TABLE | LE Supplier file |
| 14 | NULL | NULL | VA | TABLE | VA Article file |

SQLTables(hstmt,NULL,0,NULL,0,NULL,0,NULL,0)

The input parameters for SQLTables may use wildcards as for LIKE:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **TABLE\_QUALIFIER** | **TABLE\_OWNER** | **TABLE\_NAME** | **TABLE\_TYPE** | **REMARKS** |
| 1 | NULL | NULL | Currency | TABLE | Currency file |
| 2 | NULL | NULL | Customer | TABLE | ODBC Customer |
| 3 | NULL | NULL | Groups | TABLE | Article groups |
| 4 | NULL | NULL | Product | TABLE | ODBC Product |
| 5 | NULL | NULL | Suppliers | TABLE | Supplier file |

SQLTables(hstmt,NULL,0,NULL,0,"%u%",SQL\_NTS,NULL,0)

If just the qualifier is specified with % a list of valid qualifiers is returned:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **TABLE\_QUALIFIER** | **TABLE\_OWNER** | **TABLE\_NAME** | **TABLE\_TYPE** | **REMARKS** |
| 1 |  | NULL | NULL | NULL | NULL |

SQLTables(hstmt,"%",SQL\_NTS,"",0,"",0,NULL,0)

If just the owner is specified with % a list of all valid owners is returned:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **TABLE\_QUALIFIER** | **TABLE\_OWNER** | **TABLE\_NAME** | **TABLE\_TYPE** | **REMARKS** |
| 1 | NULL |  | NULL | NULL | NULL |

SQLTables(hstmt,"",0,"%",SQL\_NTS,"",0,NULL,0)

If just the tabletype is specified with % a list of valid tabletypes is returned:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **TABLE\_QUALIFIER** | **TABLE\_OWNER** | **TABLE\_NAME** | **TABLE\_TYPE** | **REMARKS** |
| 1 | NULL | NULL | NULL | SYSTEM TABLE | NULL |
| 2 | NULL | NULL | NULL | TABLE | NULL |
| 3 | NULL | NULL | NULL | TEMP | NULL |
| 4 | NULL | NULL | NULL | VIEW | NULL |

SQLTables(hstmt,"",0,"",0,"",0,"%",SQL\_NTS)

# 13. Parameters

By use of parameters in a SQL statement (?) the same select may be used for different values. SQLBindParameter assigns values for all ? in the statement.

SQLBindParameter(hstmt,1,1,SQL\_C\_CHAR,SQL\_VARCHAR,1,0,"C" ,8,&SQL\_NTS) SQLBindParameter(hstmt,2,1,SQL\_C\_CHAR,SQL\_VARCHAR,3,0,"EEE",8,&SQL\_NTS) SQLBindParameter(hstmt,3,1,SQL\_C\_LONG,SQL\_INTEGER,4,0, 30 ,4,&4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **OrderID** | **CustID** | **ProdID** | **OrderDate** | **Quantity** |
| 1 | 51 | CONSH | 72 | 1991-01-01 | 10 |
| 2 | 43 | EASTC | 69 | 1990-10-23 | 30 |
| 3 | 43 | EASTC | 71 | 1990-10-23 | 5 |
| 4 | 121 | EASTC | 60 | 1992-03-24 | 50 |

SELECT \*

FROM Orders

WHERE CustID>=? AND CustID<=? and ProdId>?

The binding can be done before or after the statement is prepared. The number of parameters may be retrieved using:

|  |  |
| --- | --- |
|  | **Number of parameters** |
| 1 | 3 |

SQLNumParams(hstmt,&params)

And a description of the parameter types may be optained by:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Type** | **Precision** | **Scale** | **Nullable** |
| 1 | 12 | 1 | 0 | 0 |
| 2 | 12 | 3 | 0 | 0 |
| 3 | 4 | 4 | 0 | 0 |

SQLDescribeParam(hstmt,\*,name,256,&len,&type,&precision,&scale,&nullable)

# 14. Parameters - Data at execution.

Parameters can be bound again with other options:

SQLBindParameter(hstmt,1,1,SQL\_C\_CHAR,SQL\_VARCHAR,1,0,1,8,SQL\_DATA\_AT\_EXEC)

The SQL\_DATA\_AT\_EXEC causes the execution of the statement to return SQL\_NEED DATA:

ERROR:SQL\_NEED\_DATA SELECT \*

FROM Orders

WHERE CustID>=? AND CustID<=? and ProdId>?

Whereafter these are transferred by repeated calls to ParamData and Putdata:

|  |  |
| --- | --- |
|  | **Parameter number** |
| 1 | SQL\_NEED\_DATA: 1 |

SQLParamData(hstmt,&nr)

*Query executed - No results returned.*

SQLPutData(hstmt,"E",SQL\_NTS);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **OrderID** | **CustID** | **ProdID** | **OrderDate** | **Quantity** |
| 1 | 43 | EASTC | 69 | 1990-10-23 | 30 |
| 2 | 43 | EASTC | 71 | 1990-10-23 | 5 |
| 3 | 121 | EASTC | 60 | 1992-03-24 | 50 |

SQLParamData(hstmt,&nr)

Unless this procedure is cancelled with SQL\_CANCEL:

*Query executed - No results returned.*

SQLCancel(hstmt)

The parameters for a statement remains active until the statement is dropped or the parameters removed with:

*Query executed - No results returned.*

SQLFreeStmt(hstmt,SQL\_RESET\_PARAMS)

# 15. Options

The following CONNECT options is used: If SQL\_ACCESS\_MODE is SQL\_MODE\_READ\_ONLY no update is possible.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Value** | **Description** |
| 1 | SQL\_ACCESS\_MODE | 0 | SQL\_MODE\_READ\_WRITE |
| 2 | SQL\_AUTOCOMMIT | 1 | SQL\_AUTOCOMMIT\_ON |
| 3 | SQL\_CURRENT\_QUALIFIER |  |  |
| 4 | SQL\_LOGIN\_TIMEOUT | 15 |  |
| 5 | SQL\_ODBC\_CURSORS | 2 | SQL\_CUR\_USE\_DRIVER |
| 6 | SQL\_OPT\_TRACE | 0 | SQL\_OPT\_TRACE\_OFF |
| 7 | SQL\_OPT\_TRACEFILE | \SQL.LOG |  |
| 8 | SQL\_PACKET\_SIZE | 1024 |  |
| 9 | SQL\_QUIET\_MODE | 0 |  |
| 10 | SQL\_TRANSLATE\_DLL |  |  |
| 11 | SQL\_TRANSLATE\_OPTION | 0 |  |
| 12 | SQL\_TXN\_ISOLATION | 0 |  |

SQLGetConnectOption(hstmt,\*,option)

The following is returned as defult STATEMENT options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Value** | **Description** |
| 1 | SQL\_ASYNC\_ENABLE | 0 | SQL\_ASYNC\_ENABLE\_OFF |
| 2 | SQL\_BIND\_TYPE | 0 | SQL\_BIND\_BY\_COLUMN |
| 3 | SQL\_CONCURRENCY | 1 | SQL\_CONCUR\_READ\_ONLY |
| 4 | SQL\_CURSOR\_TYPE | 0 | SQL\_CURSOR\_FORWARD\_ONLY |
| 5 | SQL\_KEYSET\_SIZE | 0 |  |
| 6 | SQL\_MAX\_LENGTH | 0 |  |
| 7 | SQL\_MAX\_ROWS | 0 |  |
| 8 | SQL\_NOSCAN | 0 | SQL\_NOSCAN\_OFF |
| 9 | SQL\_QUERY\_TIMEOUT | 0 |  |
| 10 | SQL\_RETRIEVE\_DATA | 1 | SQL\_RD\_ON |
| 11 | SQL\_ROWSET\_SIZE | 1 |  |
| 12 | SQL\_SIMULATE\_CURSOR | 0 | SQL\_SC\_NON\_UNIQUE |
| 13 | SQL\_USE\_BOOKMARKS | 0 | SQL\_UB\_OFF |
| 14 | SQL\_GET\_BOOKMARK | 0 | \* Invalid cursor state |
| 15 | SQL\_ROW\_NUMBER | 0 | \* Invalid cursor state |

SQLGetStmtOption(hstmt,\*,option)

If SQL\_ASYNC\_ENABLE is SQL\_ASYNC\_ON, SQLFetch may return SQL\_STILL\_EXECUTING if more than 1000 records is read during the fetch operation. Below this count is reduced to 10 by the statement option 1000.

*Query executed - No results returned.*

SQLSetStmtOption(hstmt,SQL\_ASYNC\_ENABLE,SQL\_ASYNC\_ENABLE\_ON)  *Query executed - No results returned.*

SQLSetStmtOption(hstmt,1000,10)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **OrderID** | **CustID** | **ProdID** | **OrderDate** | **Quantity** |
|  | 9 \* SQL\_STILL\_EXECUTING |  |  |  |  |
| 1 | 47 | SEVES | 52 | 1990-11-15 | 20 |
|  | 11 \* SQL\_STILL\_EXECUTING |  |  |  |  |
| 2 | 101 | ALWAO | 59 | 1991-12-12 | 15 |
| 3 | 101 | ALWAO | 77 | 1991-12-12 | 2 |
| 4 | 102 | TRADM | 30 | 1991-12-12 | 20 |
| 5 | 103 | EMPIT | 22 | 1991-12-13 | 52 |
| 6 | 103 | EMPIT | 35 | 1991-12-13 | 6 |
|  | 4 \* SQL\_STILL\_EXECUTING |  |  |  |  |

SELECT \*

FROM orders

WHERE (OrderID > 100 AND OrderID < 104) OR OrderID = 47

If SQL\_MAX\_ROWS is set a SELECT will try to not exeed this maximum.

*Query executed - No results returned.*

SQLSetStmtOption(hstmt,SQL\_MAX\_ROWS,5)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **OrderID** | **CustID** | **ProdID** | **OrderDate** | **Quantity** |
| 1 | 1 | MERRG | 25 | 1989-05-15 | 30 |
| 2 | 1 | MERRG | 40 | 1989-05-15 | 40 |
| 3 | 1 | MERRG | 59 | 1989-05-15 | 8 |
| 4 | 1 | MERRG | 64 | 1989-05-15 | 15 |
| 5 | 2 | FOODI | 31 | 1989-05-16 | 35 |

SELECT \*

FROM orders

SQL\_QUERY\_TIMEOUT detemines maximum number of seconds for executing a query

*Query executed - No results returned.*

SQLSetStmtOption(hstmt,SQL\_QUERY\_TIMEOUT,1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **OrderID** | **CustID** | **ProdID** | **OrderDate** | **Quantity** |
| ERROR: 901 S1T00 [SW-Tools][SQLEXECUTE][S1T00]Timeout expired |  |  |  |  |  |

SELECT \*

FROM orders,customer,product

WHERE product.ShipWt>777777

# 16. Functions

SQLGetfunctions returns the following values:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Level** | **Supported** | **NOT Supported** |
| 1 | Core | ALL |  |
| 2 | Level 1 | ALL |  |
| 3 | Level 2 |  |  |
| 4 | Level 2 |  |  |
| 5 | Level 2 (DM) |  |  |
| 6 | Level 2 |  |  |
| 7 | Level 2 |  |  |
| 8 | Level 2 |  |  |
| 9 | Level 2 |  |  |
| 10 | Level 2 |  |  |
| 11 | Level 2 |  |  |
| 12 | Level 2 |  |  |
| 13 | Level 2 |  |  |
| 14 | Level 2 |  |  |
| 15 | Level 2 |  |  |
| 16 | Level 2 |  |  |
| 17 | Level 2 |  |  |
| 18 | Level 2 |  |  |
| 19 | Level 2 (DM) |  |  |
| 20 | Level 2 |  |  |

SQLGetFunctions(hstmt,SQL\_API\_ALL\_FUNCTIONS,array)

# 17. SQLInfo

SQLInfo returns the following:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Value** | **Description** |
| 1 | SQL\_ACCESSIBLE\_PROCEDURES | N | May be procedures user cannot |
| 2 | SQL\_ACCESSIBLE\_TABLES | N | Not all tables may be accessed |
| 3 | SQL\_ACTIVE\_CONNECTIONS | 0 | No limit |
| 4 | SQL\_ACTIVE\_STATEMENTS | 0 | No limit |
| 5 | SQL\_ALTER\_TABLE | 3 | SQL\_AT\_ADD\_COLUMN |
|  |  |  | SQL\_AT\_DROP\_COLUMN |
| 6 | SQL\_BOOKMARK\_PERSISTENCE | 0 |  |
| 7 | SQL\_COLUMN\_ALIAS | N | Alias not supported |
| 8 | SQL\_CONCAT\_NULL\_BEHAVIOR | 1 | SQL\_CB\_NON\_NULL |
| 9 | SQL\_CONVERT\_BIGINT | 0 |  |
| 10 | SQL\_CONVERT\_BINARY | 0 |  |
| 11 | SQL\_CONVERT\_BIT | 0 |  |
| 12 | SQL\_CONVERT\_CHAR | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 13 | SQL\_CONVERT\_DATE | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 14 | SQL\_CONVERT\_DECIMAL | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 15 | SQL\_CONVERT\_DOUBLE | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 16 | SQL\_CONVERT\_FLOAT | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 17 | SQL\_CONVERT\_FUNCTIONS | 1 | SQL\_FN\_CVT\_CONVERT |
| 18 | SQL\_CONVERT\_INTEGER | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 19 | SQL\_CONVERT\_LONGVARBINARY | 0 |  |
| 20 | SQL\_CONVERT\_LONGVARCHAR | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 21 | SQL\_CONVERT\_NUMERIC | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 22 | SQL\_CONVERT\_REAL | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 23 | SQL\_CONVERT\_SMALLINT | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 24 | SQL\_CONVERT\_TIME | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 25 | SQL\_CONVERT\_TIMESTAMP | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 26 | SQL\_CONVERT\_TINYINT | 0 |  |
| 27 | SQL\_CONVERT\_VARBINARY | 0 |  |
| 28 | SQL\_CONVERT\_VARCHAR | 230399 | SQL\_CVT\_CHAR |
|  |  |  | SQL\_CVT\_NUMERIC |
|  |  |  | SQL\_CVT\_DECIMAL |
|  |  |  | SQL\_CVT\_INTEGER |
|  |  |  | SQL\_CVT\_SMALLINT |
|  |  |  | SQL\_CVT\_FLOAT |
|  |  |  | SQL\_CVT\_REAL |
|  |  |  | SQL\_CVT\_DOUBLE |
|  |  |  | SQL\_CVT\_VARCHAR |
|  |  |  | SQL\_CVT\_LONGVARCHAR |
|  |  |  | SQL\_CVT\_DATE |
|  |  |  | SQL\_CVT\_TIME |
|  |  |  | SQL\_CVT\_TIMESTAMP |
| 29 | SQL\_CORRELATION\_NAME | 2 | SQL\_CN\_ANY |
| 30 | SQL\_CURSOR\_COMMIT\_BEHAVIOR | 2 | SQL\_CB\_PRESERVE |
| 31 | SQL\_CURSOR\_ROLLBACK\_BEHAVIOR | 2 | SQL\_CB\_PRESERVE |
| 32 | SQL\_DATA\_SOURCE\_NAME | SWTools32 |  |
| 33 | SQL\_DATA\_SOURCE\_READ\_ONLY | N | Read/Write |
| 34 | SQL\_DATABASE\_NAME |  |  |
| 35 | SQL\_DBMS\_NAME | SW-Tools |  |
| 36 | SQL\_DBMS\_VER | 08.12.0011 |  |
| 37 | SQL\_DEFAULT\_TXN\_ISOLATION | 0 |  |
| 38 | SQL\_DRIVER\_HDBC | 16924556 |  |
| 39 | SQL\_DRIVER\_HENV | 16834748 |  |
| 40 | SQL\_DRIVER\_HLIB | 79691776 |  |
| 41 | SQL\_DRIVER\_HSTMT | 0 | \* Invalid argument value |
| 42 | SQL\_DRIVER\_NAME | SWODBC32.dll |  |
| 43 | SQL\_DRIVER\_ODBC\_VER | 02.10 |  |
| 44 | SQL\_DRIVER\_VER | 08.12.0011 |  |
| 45 | SQL\_EXPRESSIONS\_IN\_ORDERBY | Y | Yes, Expressions in orderby |
| 46 | SQL\_FETCH\_DIRECTION | 1 | SQL\_FD\_FETCH\_NEXT |
| 47 | SQL\_FILE\_USAGE | 0 | SQL\_FILE\_NOT\_SUPPORTED |
| 48 | SQL\_GETDATA\_EXTENSIONS | 11 | SQL\_GD\_ANY\_COLUMN |
|  |  |  | SQL\_GD\_ANY\_ORDER |
|  |  |  | SQL\_GD\_BOUND |
| 49 | SQL\_GROUP\_BY | 3 | SQL\_GB\_GROUP\_BY\_EQUALS\_SELECT |
|  |  |  | SQL\_GB\_GROUP\_BY\_CONTAINS\_SELEC |
| 50 | SQL\_IDENTIFIER\_CASE | 4 | SQL\_IC\_SENSITIVE |
| 51 | SQL\_IDENTIFIER\_QUOTE\_CHAR | ' |  |
| 52 | SQL\_KEYWORDS | UPPER,EXPORT,IMPORT |  |
| 53 | SQL\_LIKE\_ESCAPE\_CLAUSE | Y | LIKE fully supported |
| 54 | SQL\_LOCK\_TYPES | 0 |  |
| 55 | SQL\_MAX\_BINARY\_LITERAL\_LEN | 64 |  |
| 56 | SQL\_MAX\_CHAR\_LITERAL\_LEN | 64 |  |
| 57 | SQL\_MAX\_COLUMNS\_IN\_GROUP\_BY | 499 |  |
| 58 | SQL\_MAX\_COLUMNS\_IN\_INDEX | 499 |  |
| 59 | SQL\_MAX\_COLUMNS\_IN\_ORDER\_BY | 499 |  |
| 60 | SQL\_MAX\_COLUMNS\_IN\_SELECT | 499 |  |
| 61 | SQL\_MAX\_COLUMNS\_IN\_TABLE | 499 |  |
| 62 | SQL\_MAX\_COLUMN\_NAME\_LEN | 64 |  |
| 63 | SQL\_MAX\_CURSOR\_NAME\_LEN | 18 |  |
| 64 | SQL\_MAX\_INDEX\_SIZE | 499 |  |
| 65 | SQL\_MAX\_OWNER\_NAME\_LEN | 0 | No limit |
| 66 | SQL\_MAX\_PROCEDURE\_NAME\_LEN | 0 | No limit |
| 67 | SQL\_MAX\_QUALIFIER\_NAME\_LEN | 0 | No limit |
| 68 | SQL\_MAX\_ROW\_SIZE | 0 | No limit |
| 69 | SQL\_MAX\_ROW\_SIZE\_INCLUDES\_LONG | Y | SQL\_LONGVARCHAR included in MA |
| 70 | SQL\_MAX\_STATEMENT\_LEN | 0 | No limit |
| 71 | SQL\_MAX\_TABLE\_NAME\_LEN | 64 |  |
| 72 | SQL\_MAX\_TABLES\_IN\_SELECT | 64 |  |
| 73 | SQL\_MAX\_USER\_NAME\_LEN | 64 |  |
| 74 | SQL\_MULT\_RESULT\_SETS | N | No support |
| 75 | SQL\_MULTIPLE\_ACTIVE\_TXN | Y | More conn.with active trans |
| 76 | SQL\_NEED\_LONG\_DATA\_LEN | N | No need for long data length |
| 77 | SQL\_NON\_NULLABLE\_COLUMNS | 0 | SQL\_NCC\_NULL |
| 78 | SQL\_NULL\_COLLATION | 1 | SQL\_NC\_LOW |
| 79 | SQL\_NUMERIC\_FUNCTIONS | 16777215 | SQL\_FN\_NUM\_ABS |
|  |  |  | SQL\_FN\_NUM\_ACOS |
|  |  |  | SQL\_FN\_NUM\_ASIN |
|  |  |  | SQL\_FN\_NUM\_ATAN |
|  |  |  | SQL\_FN\_NUM\_ATAN2 |
|  |  |  | SQL\_FN\_NUM\_CEILING |
|  |  |  | SQL\_FN\_NUM\_COS |
|  |  |  | SQL\_FN\_NUM\_COT |
|  |  |  | SQL\_FN\_NUM\_EXP |
|  |  |  | SQL\_FN\_NUM\_FLOOR |
|  |  |  | SQL\_FN\_NUM\_LOG |
|  |  |  | SQL\_FN\_NUM\_MOD |
|  |  |  | SQL\_FN\_NUM\_SIGN |
|  |  |  | SQL\_FN\_NUM\_SIN |
|  |  |  | SQL\_FN\_NUM\_SQRT |
|  |  |  | SQL\_FN\_NUM\_TAN |
|  |  |  | SQL\_FN\_NUM\_PI |
|  |  |  | SQL\_FN\_NUM\_RAND |
|  |  |  | SQL\_FN\_NUM\_DEGREES |
|  |  |  | SQL\_FN\_NUM\_LOG10 |
|  |  |  | SQL\_FN\_NUM\_POWER |
|  |  |  | SQL\_FN\_NUM\_RADIANS |
|  |  |  | SQL\_FN\_NUM\_ROUND |
|  |  |  | SQL\_FN\_NUM\_TRUNCATE |
| 80 | SQL\_ODBC\_API\_CONFORMANCE | 1 | SQL\_OAC\_LEVEL1 |
| 81 | SQL\_ODBC\_SAG\_CLI\_CONFORMANCE | 1 | SQL\_OSCC\_COMPLIANT |
| 82 | SQL\_ODBC\_SQL\_CONFORMANCE | 1 | SQL\_OSC\_CORE |
| 83 | SQL\_ODBC\_SQL\_OPT\_IEF | N | No Optional Integrity Enhancem |
| 84 | SQL\_ODBC\_VER | 03.51.0000 |  |
| 85 | SQL\_ORDER\_BY\_COLUMNS\_IN\_SELECT | N | ORDER BY free |
| 86 | SQL\_OUTER\_JOINS | Y | OUTER JOINS Supported |
| 87 | SQL\_SQL\_OJ\_CAPABILITIES | 1 | SQL\_OJ\_LEFT |
| 88 | SQL\_OWNER\_TERM | OWNER |  |
| 89 | SQL\_OWNER\_USAGE | 0 |  |
| 90 | SQL\_POS\_OPERATIONS | 0 |  |
| 91 | SQL\_POSITIONED\_STATEMENTS | 7 | SQL\_PS\_POSITIONED\_DELETE |
|  |  |  | SQL\_PS\_POSITIONED\_UPDATE |
|  |  |  | SQL\_PS\_SELECT\_FOR\_UPDATE |
| 92 | SQL\_PROCEDURE\_TERM | PROCEDURE |  |
| 93 | SQL\_PROCEDURES | N | Procedures NOT supported |
| 94 | SQL\_QUALIFIER\_LOCATION | 1 | SQL\_QL\_START |
| 95 | SQL\_QUALIFIER\_NAME\_SEPARATOR | \ |  |
| 96 | SQL\_QUALIFIER\_TERM | DIRECTORY |  |
| 97 | SQL\_QUALIFIER\_USAGE | 0 |  |
| 98 | SQL\_QUOTED\_IDENTIFIER\_CASE | 4 | SQL\_IC\_SENSITIVE |
| 99 | SQL\_ROW\_UPDATES | N | No |
| 100 | SQL\_SCROLL\_CONCURRENCY | 0 |  |
| 101 | SQL\_SCROLL\_OPTIONS | 0 |  |
| 102 | SQL\_SEARCH\_PATTERN\_ESCAPE | \ |  |
| 103 | SQL\_SERVER\_NAME | SW-Tools |  |
| 104 | SQL\_SPECIAL\_CHARACTERS | #ÆØÅæøåÄÖÜäöüß |  |
| 105 | SQL\_STATIC\_SENSITIVITY | 0 |  |
| 106 | SQL\_STRING\_FUNCTIONS | 294911 | SQL\_FN\_STR\_CONCAT |
|  |  |  | SQL\_FN\_STR\_INSERT |
|  |  |  | SQL\_FN\_STR\_LEFT |
|  |  |  | SQL\_FN\_STR\_LTRIM |
|  |  |  | SQL\_FN\_STR\_LENGTH |
|  |  |  | SQL\_FN\_STR\_LOCATE |
|  |  |  | SQL\_FN\_STR\_LCASE |
|  |  |  | SQL\_FN\_STR\_REPEAT |
|  |  |  | SQL\_FN\_STR\_REPLACE |
|  |  |  | SQL\_FN\_STR\_RIGHT |
|  |  |  | SQL\_FN\_STR\_RTRIM |
|  |  |  | SQL\_FN\_STR\_SUBSTRING |
|  |  |  | SQL\_FN\_STR\_UCASE |
|  |  |  | SQL\_FN\_STR\_ASCII |
|  |  |  | SQL\_FN\_STR\_CHAR |
|  |  |  | SQL\_FN\_STR\_SPACE |
| 107 | SQL\_SUBQUERIES | 31 | SQL\_SQ\_COMPARISON |
|  |  |  | SQL\_SQ\_EXISTS |
|  |  |  | SQL\_SQ\_IN |
|  |  |  | SQL\_SQ\_QUANTIFIED |
|  |  |  | SQL\_SQ\_CORRELATED\_SUBQUERIES |
| 108 | SQL\_SYSTEM\_FUNCTIONS | 7 | SQL\_FN\_SYS\_USERNAME |
|  |  |  | SQL\_FN\_SYS\_DBNAME |
|  |  |  | SQL\_FN\_SYS\_IFNULL |
| 109 | SQL\_TABLE\_TERM | TABLE |  |
| 110 | SQL\_TIMEDATE\_ADD\_INTERVALS | 0 |  |
| 111 | SQL\_TIMEDATE\_DIFF\_INTERVALS | 0 |  |
| 112 | SQL\_TIMEDATE\_FUNCTIONS | 106495 | SQL\_FN\_TD\_NOW |
|  |  |  | SQL\_FN\_TD\_CURDATE |
|  |  |  | SQL\_FN\_TD\_DAYOFMONTH |
|  |  |  | SQL\_FN\_TD\_DAYOFWEEK |
|  |  |  | SQL\_FN\_TD\_DAYOFYEAR |
|  |  |  | SQL\_FN\_TD\_MONTH |
|  |  |  | SQL\_FN\_TD\_QUARTER |
|  |  |  | SQL\_FN\_TD\_WEEK |
|  |  |  | SQL\_FN\_TD\_YEAR |
|  |  |  | SQL\_FN\_TD\_CURTIME |
|  |  |  | SQL\_FN\_TD\_HOUR |
|  |  |  | SQL\_FN\_TD\_MINUTE |
|  |  |  | SQL\_FN\_TD\_SECOND |
|  |  |  | SQL\_FN\_TD\_DAYNAME |
|  |  |  | SQL\_FN\_TD\_MONTHNAME |
| 113 | SQL\_TXN\_CAPABLE | 0 | SQL\_TC\_NONE |
| 114 | SQL\_TXN\_ISOLATION\_OPTION | 0 |  |
| 115 | SQL\_UNION | 3 | SQL\_U\_UNION |
|  |  |  | SQL\_U\_UNION\_ALL |
| 116 | SQL\_USER\_NAME |  |  |

SQLInfo(hstmt,\*)

# Figure list

1. Simple SELECT 8

2. Using ORDER BY 9

3. ODBC on BASIC files with simple WHERE clause 9

4. Using calculations 10

5. Special column names encloded in quotes 11

6. Using AS clause for correlation name of table 13

7. Using OUTER JOIN 14

8. Using LEFT OUTER JOIN 14

9. Multiple SELECTs for subqueries 15

10. Using comparision operators 15

11. Sample use of ANY comparision 15

12. Scanning result set when using IN clause 16

13. Aggregate functions COUNT, SUM, MAX, MIN, AVG 17

14. Using the same table multiple times 17

15. Using the LIKE function 18

16. Using the MATCHES function 18

17. Calling functions within SELECT 19

18. Calling functions within SELECT with full ODBC syntax 19

19. Using Date, Time and Timestamp syntax 20

20. Using standard SQL syntax for dates 20

21. Using field numbers instead of field names 21

22. Subscribed fields 22

23. A simple GROUP BY sample 23

24. A simple HAVING sample 23

25. SELECT using DISTINCT 23

26. SELECT using DISTINCT on aggregate functions 24

27. SELECT using UNIONs 24

28. VALUES constructor 25

29. SELECT from result set 25

30. Joined tables without where 25

# Index

A

ABS 7;64

ACCESS 5;57

Administrator 4

API-Level 3

ASIN 7;65

ATAN 7;65

ATAN2 7;65

AVG 17;68

B

BETWEEN 10

C

Calculations 10

CEILING 7;65

CHAR 7;46;47;48;55;56;60;61;62;63;64;66

Company 6

CONCAT 7;60;66

CONV 7

CONVERT 19;60;61;62;63

Correlation 13

COS 7;65

COT 7;65

COUNT 17;23;24;68

CURDATE 7;66

CURTIME 7;66

D

DATE 7;48;49;50;60;61;62;63

DAY 7

DAYNAME 7;66

DAYOFMONTH 7;66

DAYOFWEEK 7;66

DAYOFYEAR 7;66

DDMMYY 20

DEGREES 7;65

DESC 8;9;46

DISTINCT 23;24;68

E

EDIT 7

EXISTS 15;66

EXP 7;65

EXPR-1 10;17;19;22;23

F

FIND 7

FLOOR 7;65

FNA 7

FNB 7

FND 7

FNE 7

FNF 7

FNH 7

FNO 7

FNR 7

FNU 7

FNV 7

FNY 7

G

GROUP 23;26;64;68

H

HAVING 23;68

HOUR 7;66

I

IN 7;11;12;16;51;63;64;65;66;68

INI 6;52;53

INSERT 7;28;29;30;66

Installation 4

INT 7

ISNULL 7

J

JOINS 14;65

L

LCASE 7;19;66

LEFT 7;14;65;66;68

LEN 7;49;50;64

LENGTH 7;48;49;57;66

LIKE 7;18;54;64;68

LOCATE 7;66

LOG 7;57;65

LOG10 7;65

LOWER 7

LTRIM 7;66

M

MATCHES 7;18;68

MAX 17;48;49;57;58;64;68

MIN 17;48;68

MINUTE 7;66

MOD 7;65

MONTH 7;66

MONTHNAME 7;66

N

NAME 7;48;49;50;52;53;54;63;64;65;66;67

NOT 7;10;18;59;64;65

NULL 6;48;49;50;51;52;53;54;60;64

NUMBER 7;21;57

O

ON 14;35;46;57

ONLY 57;63

ORDER 8;9;21;24;26;27;37;38;64;65;68

OUTER 14;41;65;68

Owner 6;53

P

PI 7;65

POW 7

POWER 7;65

Q

Qualifier 6;53

QUARTER 7;66

R

RADIANS 7;65

RAND 7;65

RECNO 21

REPEAT 7;66

REPLACE 7;66

RIGTH 7

ROUND 7;65

RTRIM 7;66

RUN 7

RUND 7

S

SECOND 7;66

SELECT 8;9;10;11;12;13;14;15;16;17;18;19;20;21;22;23;24;25;27;28;29;30;31;33;34;35;37;38;39;41;42;43;44;45;49;55;56;58;64;65;68

SETUP 4

SGN 7

SIGN 7;65

SIN 7;65

SMAA 7

SOGE 7

SOME 15

SPACE 7;66

SPOFF 7

SQL\_INTEGER 19;49;55

SQLDate 20

SQR 7

SQRT 7;65

Subqueries 15

SUBSTRING 7;66

SUM 17;23;24;68

T

TABLE 21;26;36;45;46;47;49;50;51;52;53;54;60;64;66

TAN 7;65

TIME 7;48;60;61;62;63

Timestamp 20;68

TOCHAR 7

TODBL 7

TOLONG 7

TOSHORT 7

TRUNCATE 7;65

U

UCASE 7;66

UNION 23;24;67

UNIONs 24;68

Update 6;42

UPPER 7;64

USER 7;64;67

V

VALCH 7

VALID 7

VALUES 24;25;28;29;68

W

WDAY 7

WEEK 7;66

WHERE 9;10;11;12;13;14;15;16;17;18;20;21;24;25;29;30;32;34;41;42;43;44;49;55;56;58;68

WORD 5

Y

YEAR 7;66

YYMMDD 20

YYYY-MM-DD 20

YYYYMMDDHHMMSS 20