English User Manual

Phone:	+45) 33 33 05 56
Mail:	swtools@swtools.com
www:	www.swtools.com

SW-Tools ODBC - Programmers Reference

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

Contents

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	CREDIT CARD	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	CREDIT CARD	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	10000	60000	1993-	123	2	1	100	0

			0		12-15					
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	MACHI	2000	1500	1994-	100	1	10	0	0

		NE			06-01					
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	CREDIT CARD	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	CREDIT CARD	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	TIM	11	19	#	#	NU	1	0	2	NU	0	NU	NU	NU	NU

	ESTAMP					LL				L		LL	L	L	L
12	VARCHAR	12	1024	"	"	MAX LENGTH	0	1	3	NULL	0	NULL	NULL	NULL	NULL

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

Index