Mimer SQL

Features and Functionality History

Produced and published by Mimer InformationTechnology AB, Uppsala, Sweden.
P.O. Box 1713,
SE-751 47 Uppsala, Sweden.
Tel +46(0)18-780 92 00.
Fax +46(0)18-780 92 40.

Mimer Web Sites:
http://developer.mimer.com
http://www.mimer.com
http://www.upright.se

Contents

Chapter 1

New Features and Functions

This chapter describes new features and functionality introduced into Mimer SQL in previous versions.

The Mimer SQL Database Server

Performance (V9.2.1)

A large number of performance improvements have been made in this version.

The improvements include the introduction of a compilation information cache which significantly reduces compilation time.

LOAD/UNLOAD (V9.1.1)

New commands, `LOAD` and `UNLOAD`, have been implemented to facilitate transferring data to and from databases. `LOAD` and `UNLOAD` can be used with Mimer SQL from any ODBC-based SQL command interpreter. For more information, see the *Mimer SQL System Management Handbook.*

PSM Debugger (V8.2.4)

A Java-based graphic debugger for PSM routines has been added. The PSM Debugger has support for watching variables, step-wise execution and setting breakpoints. You can debug procedures, functions and triggers.

The PSM Debugger requires a Java 2 (version 1.2 or later) compatible Java runtime environment. For more information, see the *Mimer SQL Programmer's Manual*.

Start the debugger with this command:

```
java -jar psmdebug.jar
```

The syntax for the database URL in the login dialog box is:

```
hostname[:port]/database
```

If the database resides on your local machine, specify `localhost` as the host name.

Examples:

```
localhost/testdb

my_node.mimer.se/supplier

my_node.mimer.se:1365/supplier_temp
```

Mimer SQL Structured Query Language

SUBSTRING on CLOB and NCLOB Columns (V9.2.5)

The SUBSTRING function can now be used on columns with data type CHARACTER LARGE OBJECT or NATIONAL CHARACTER LARGE OBJECT.

Example:

```
SELECT column1,
       SUBSTRING(clobcolumn FROM 1 FOR 50) AS intro
FROM sometable;
```

Using LIKE Predicate for CLOB and NCLOB Columns (V9.2.4)

The LIKE predicate can now be used to search the contents of columns with data type CHARACTER LARGE OBJECT or NATIONAL CHARACTER LARGE OBJECT.

Example:

```
SELECT * FROM tab1 WHERE clobcol LIKE '%Mimer%Uppsala%';
```

ORDER BY on Expressions (V9.2.4)

The ORDER BY clause can now contain expressions. An expression in the ORDER BY specification must contain at least one column reference to a table in the query.

Examples:

```
SELECT column1 FROM sometable ORDER BY column2;
SELECT column1 FROM sometable ORDER BY ABS(column1);
SELECT column1 FROM sometable
    ORDER BY CASE WHEN column1 < column2 THEN column1 ELSE column2 END;
table1 JOIN table2 USING (column1) ORDER BY somefunction(column1);
```

See *Mimer SQL Reference Manual*, Chapter 12 for more information.

INSERT Large Object Column With a Subselect (V9.2.3)

A column with a Large Object data type can now participate in INSERT statements with subselect, i.e. the values to be inserted can be fetched from another table in the database.

Example:

```
INSERT INTO tab2 SELECT c1, lobcol FROM tab1 WHERE c1 < 10;
```

However, a Large Object can still not be used in an UPDATE with subselect statement

Extended Support for Collations (V9.2.1)

Collations for all European languages are supported. (See list in *Mimer SQL Reference Manual, Chapter 4*.)

It is also possible to create new collations based on existing collations for specific purposes. See the updated `CREATE COLLATION` statement in the reference manual.

New Scalar Functions (V9.2.1)

Two new functions for converting between a Unicode character and a numerical value has been introduced. The function `UNICODE_CODE` will given a Unicode character return the corresponding numerical value and the `UNICODE_CHAR` function returns a Unicode character.

Examples:

```
SET uc = UNICODE_CHAR(4711);
SET i = UNICODE_CODE(n'Ö');
```

New Views in INFORMATION_SCHEMA (V9.2.1)

The new views are `INFORMATION_SCHEMA.SQL_FEATURES`, `INFORMATION_SCHEMA.SQL_SIZING`, and `INFORMATION_SCHEMA.EXT_COLLATION_DEFINITIONS`.

See *Mimer SQL Reference Manual, Chapter 13* for more information on these views.

Support for Scalar Subqueries (V9.2.1)

A scalar subquery can be used in a DML statement wherever an expression can be used.

Examples:

```
CREATE FUNCTION f() RETURNS int READS SQL DATA
    RETURN (SELECT count(*) FROM mimer_store.music)

SELECT title,
      (SELECT title FROM tracks
       WHERE albumID = a.albumID
       AND seqno = 1) AS track
FROM album AS a
```

WITH HOLD (V9.2.1)

Support for `WITH HOLD` cursors have been added. Cursors opened `WITH HOLD` are not implicitly closed by `COMMIT`. For more information, see the *Mimer SQL Reference Manual.*

NCLOB Data Type (V9.2.1)

The `NCLOB` (`NATIONAL CHARACTER LARGE OBJECT`) data type is now supported.

ALTER STATEMENT (V9.2.1)

The `ALTER STATEMENT` command can be used to get a new execution plan for a precompiled statement. This is useful when, for instance, an index has been added or the statistics data has been updated.

Example

```
ALTER STATEMENT book_statement_stmt REFRESH;
```

Triggers (V8.2.1)

A trigger is a stored procedure that is executed every time a data manipulation statement is executed. A trigger is associated with an `INSERT`, `UPDATE` or `DELETE` operation on a certain table. In the trigger almost all PSM statements may be used. In a trigger, it is possible to reject the operation that caused the trigger to execute.

The trigger is executed once for each statement, either before or after the statement is executed. For a complete description of this concept, see the *Mimer SQL Reference Manual.*

Instead-of Triggers (V8.2.1)

An instead-of trigger is a trigger that is executed instead of performing the actual operation. Instead-of triggers are used with views only. It is possible to make join-views ‘updateable’ by using instead-of triggers for the different operations. In such cases, it is expected that the view will modify one or more other tables to simulate the data manipulation function on the view.

For a complete description of this command, see the *Mimer SQL Reference Manual.*

PSM Functions (V8.2.1)

PSM functions are particularly useful because they can be referenced from an SQL statement. This allows procedural code to be activated within an SQL statement.

All parameters to a function are input and the function value is the only value returned from the function. A deterministic function always returns the same value for a given set of input parameters. Otherwise, it is a non-deterministic function. Non-deterministic functions are used by the optimizer to determine possible evaluation orders. In the future, it will be possible to build secondary indexes on deterministic functions.

Schema Support (V8.2.1)

It is now possible to have an object owner without having a corresponding `USER` or `PROGRAM`. The term used is schema.

To use the `CREATE SCHEMA` statement the ident must have `SCHEMA` or `IDENT` privilege. `SYSADM` is given `SCHEMA` privilege with grant option.

A user may have zero, one or more schemas. By default, a schema with the same name as the user is created when the ident is created.

Objects are created in a schema by specifying the schema name in the `CREATE` statement:

Example:

```
As user SYSADM:
GRANT SCHEMA TO LOKADM;

As user LOKADM:
CREATE SCHEMA LOKUS;
CREATE TABLE LOKUS.CUSTOMER(CUSTID INTEGER,CUST_NAME CHAR(30));
CREATE VIEW LOKUS.MAJOR_CUSTOMER AS ...

DROP SCHEMA LOKUS CASCADE;
```

Sequences (V8.2.1)

A new construct called a `SEQUENCE` has been implemented. A `SEQUENCE` returns unique integer values regardless of concurrent access to the system. It is also possible to retrieve the previous value returned to the application.

A `SEQUENCE` can be defined as being unique or not. If the database server is improperly shut down, there may be a gap in the `SEQUENCE`. That is, the system will always return unique values for a unique `SEQUENCE`, but after an improper system stop a set of values may be skipped.

A `SEQUENCE` can be used as a default value of a column or domain. For example:

```
CREATE UNIQUE SEQUENCE SEQ3;

CREATE TABLE TAB3(C1 INT DEFAULT NEXT_VALUE OF SEQ3);
```

Relaxed Rules for Views (V8.2.1)

It is now permitted to have a query-expression in a view. This makes it possible to create views that use `UNION` and/or `UNION ALL`.

Furthermore, it is now permitted to use views that contain `GROUP BY` and/or `HAVING` (so called grouped views) in any manner. For example, it is possible to perform a second grouping in the query if wanted.

Updateable Primary Keys (V8.2.1)

The system now allows primary keys to be updated. Previously, the application did not have privilege to update primary keys.

These privileges have now been given to the user. The operation is only allowed if the table is located in a databank with `TRANS` or `LOG` option.

Named Constraints (V8.2.1)

Constraints may now be given a name. If no name is given the system will generate a name. Constraint names can be viewed using the following INFORMATION_SCHEMA views:

- TABLE_CONSTRAINTS
- REFERENTIAL_CONSTRAINTS
- CHECK_CONSTRAINTS

Examples:

```
CREATE TABLE ACCOUNT( AMOUNT DECIMAL(15,2) CONSTRAINT ZERO_CHECK CHECK(AMOUNT
<> 0))

ALTER TABLE ACCOUNT DROP CONSTRAINT ZERO_CHECK
```

Recursive Procedures (V8.2.1)

Procedures and functions may now be called recursively. The currently permitted nesting level is 42.

Record Data Type (V8.2.1)

A new record data type, for use in PSM, has been introduced.

Example:

```
DECLARE VAR3 ROW(FIELD1 INT, FIELD2 VARCHAR(22));
DECLARE TTYP1 ROW AS (TAB1);
DECLARE TTYP2 ROW AS (TAB1(COL2, COL3));

SET TTYP1.COL3 = TTYP2.COL3;
```

ATOMIC (V8.2.1)

The ATOMIC declaration statement allows a number of PSM statements to be grouped together. If a statement in the group fails, statements that affect the database are undone.

An atomic may also be combined with an UNDO exception handler. See the next section.

UNDO (V8.2.1)

An exception handler in a compound statement can be declared as having the type undo.

This means that if the handler is chosen as the most appropriate handler for a certain exception, all database operations that are done in the compound statement that contains the handler declaration will be undone before the handler code is executed.

128 Character Names (V8.2.1)

The maximum length of identifiers in Mimer SQL has been changed from 18 to 128 characters. Objects such as idents, table names, etc. can be up to 128 characters. Passwords are limited to 18 characters.

System View Support (V8.2.1)

Four new schemas are available for looking up information about the system.

`INFORMATION_SCHEMA` – This is the new set of standard system views in Mimer SQL. They contain a set of views that are implemented in accordance with the SQL92 standard and a set that can be used for retrieving information about Mimer SQL specific objects e.g. databanks and sequences.

`INFO_SCHEM` – This is a variant of the INFORMATION_SCHEMA view which uses only 18 character column names in the views. Whenever possible, the INFORMATION_SCHEMA views should be used instead.

`FIPS_DOCUMENTATION` – These views document Mimer SQL's conformance to the SQL2 standard. The schema also includes a view which provides a number of limits which may be tested for at runtime.

`OLD_INFO_SCHEMA` – The views in `OLD_INFO_SCHEMA` replaces the X/Open 1992 standard tables according to X/Open SQL standard for 1995. They should only be used if you have previously used the INFORMATION_SCHEMA views from X/Open 1992. These views are not created automatically when the system is created but can be defined manually.

The Reference manual contains descriptions of the views that are present in each schema. Currently this information is somewhat outdated.

To see which views that are defined and which columns they contain, the views `information_schema.views` and `information_schema.columns` can be used.

Like Expression (V8.2.1)

A like pattern may now be an expression. For example:

```
SELECT * FROM TAB1 WHERE COL1 LIKE ? || '%';
```

In the example, the question mark is a host variable supplied by the application. The query effectively implements a begins with predicate (assuming the host variable contains no percent or underscore characters).

Soundex Function (V8.2.1)

The soundex function returns a string expression that is the same for strings that have the same pronunciation. This is, of course, language dependent. Check how the current algorithm works for your language, and send feedback to your Mimer SQL distributor.

Example:

```
SELECT * FROM TAB1
WHERE SOUNDEX(NAME) = SOUNDEX('Karlson');
```

New Reserved Words (V8.2.1)

The following keywords have been added to the list of reserved words:

```
ARE, AT, AUTHORIZATION, CASCADED, CAST, CHARACTER, CHECK, COLLATE,
COLUMN, CONSTRAINT, CORRESPONDING, CROSS, CURRENT_PATH, CURRENT_USER,
DAY, DEFAULT, EACH, EXCEPT, FALSE, FOREIGN, FULL, FUNCTION, GLOBAL, HOLD,
HOUR, IDENTITY, INDICATOR, INTERSECT, LARGE, LOCAL, LOCALTIME,
LOCALTIMESTAMP, MATCH, MINUTE, MONTH, NATIONAL, NEW, OLD, ONLY,
OVERLAPS, PRECISION, PRIMARY, REFERENCING, RELEASE, ROW, SECOND,
SESSION_USER, SPECIFIC, SYSTEM_USER, TABLE, TIMEZONE_HOUR,
TIMEZONE_MINUTE, TRANSLATION, TRUE, UNIQUE, UNKNOWN, VALUE, VARYING,
WITHOUT, YEAR.
```

The following word is no longer reserved:

```
KEY
```

SESSION_USER (V8.2.1)

The `SESSION_USER` function returns the name that was used when connecting to Mimer. In most cases, the value of `CURRENT_USER` will be the same as `SESSION_USER`. The only difference is that when used in a routine or trigger, the value of `CURRENT_USER` will be the name of the ident that created the routine or trigger being executed.

UNIQUE Columns May Be NULL (V8.2.1)

Previously, columns that were part of a UNIQUE key were required to be declared as `NOT NULL`. This restriction has been removed.

If any value of a `UNIQUE` contains a `NULL` value, the value is accepted. That is, duplicate values may exist as long as at least one of the columns contains a `NULL` value. This is different from a `UNIQUE INDEX` which only allows a single `NULL` to exist.

OVERLAPS (V8.2.1)

An overlap predicate tests two ranges of dates and times to see if the ranges overlap.

Example

```
SELECT * FROM BOOK_GUEST WHERE (ARRIVE,DEPART) OVERLAPS
(date'2001-02-01',interval '10' day);
```

GRANT/REVOKE on Individual Column (V8.2.1)

The system now allows `GRANT` and `REVOKE` on individual columns. This is allowed for `INSERT`, `UPDATE` and `REFERENCES` privileges. For example:

```
GRANT REFERENCES(COL1,COL2) ON TABLE T TO USER1,USER2
WITH GRANT OPTION;

REVOKE INSERT(ORDER_NUMBER) ON ACCOUNT FROM MANAGER;
```

Revoke GRANT OPTION (V8.2.1)

It is now possible to only remove the `GRANT OPTION` without revoking the access privilege. Continuing the example from the previous section:

```
REVOKE GRANT OPTION FOR REFERENCES(COL2) ON TABLE T FROM USER1;
```

COMMENT ON New Objects (V8.2.1)

`COMMENT ON` now supports the following object types:

- `FUNCTION`
- `SEQUENCE`
- `TRIGGER`
- `SCHEMA`

Optional AS in Select List (V8.2.1)

The keyword `AS` is no longer mandatory in the select list. For clarity, it is recommended that the keyword `AS` be retained. For example:

```
SELECT COL1 + COL2 AS CCSUM, COL3 FROM T;
SELECT COL1 + COL2 CCSUM, COL3 FROM T;
```

Explicit Defaults (V8.2.1)

The keyword `DEFAULT` may be used in `INSERT` and `UPDATE` statements to force the system to store the current default value for the column. For example:

```
INSERT INTO TAB2(C1,C2) VALUES (DEFAULT, 'ABC');
UPDATE TAB2 SET C2 = DEFAULT WHERE C1 = 10;
```

LOCALTIME, LOCALTIMESTAMP (V8.2.1)

Support for the functions `LOCALTIME` and `LOCALTIMESTAMP` has been added. These functions return the `CURRENT_TIME` and `CURRENT_TIMESTAMP` without time zone displacement.

For example:

```
CREATE TABLE LOG_ATTEMPTS(USER_NAME VARCHAR(128),
LOG_TIMESTAMP TIMESTAMP DEFAULT
LOCALTIMESTAMP, …
```

Currently, these functions have the same functionality as the `CURRENT_TIME` and `CURRENT_TIMESTAMP` functions. It is preferable that the new functions are used, as `CURRENT_TIME` and `CURRENT_TIMESTAMP` will, in future versions, include a time zone displacement.

SET Statement (V8.2.1)

The `SET` statement, for assigning a value to a variable, can now be used outside of routines. For example:

```
SET ? = 100*378.50+1800*178+285*177+584*126;
```

Binary Data Type (V8.2.1)

A binary data type is supported. See the *Mimer SQL Reference Manual* for a more information.

Alter Table (V8.2.1)

It is possible to add and drop constraints from a table. For example:

```
ALTER TABLE ACCOUNT DROP CONSTRAINT CUSTOMER_NUMBER;

ALTER TABLE ACCOUNT ADD CONSTRAINT CUSTOMER_LOOKUP FOREIGN
KEY(CUSTOMER_NUMBER) REFERENCES CUSTOMER;
```

The data type of a column may be changed as long as the new type is assignment compatible with the previous data type. E.g. it is possible to change from integer to decimal and also from character to character varying. For example:

```
ALTER TABLE ORDER_LINE ALTER COLUMN AMOUNT DECIMAL(15,2);

ALTER TABLE CUSTOMER ALTER COLUMN CUSTOMER_NAME VARCHAR(40);
```

On Delete Rule (V8.2.1)

When defining a foreign key reference, it is possible to declare a delete rule that defines what action will take place when a row in the referenced table is deleted. The following actions can be specified: `NO ACTION`, `SET NULL`, `SET DEFAULT` and `CASCADE`. For example:

```
ALTER TABLE ACCOUNT ADD CONSTRAINT CUSTOMER_LOOKUP FOREIGN
KEY(CUSTOMER_NUMBER) REFERENCES CUSTOMER
ON DELETE CASCADE;
```

See the *Mimer SQL Reference Manual* for a description of the different actions.

Get Diagnostics in PSM (V8.2.1)

It is now possible to use all variants of the `GET DIAGNOSTICS` statement in routines and triggers.

Compiler Directives (V8.2.1)

An SQL statement may contain directives to the SQL compiler. With these directives, it is possible to affect the access path for a query or whether an index should be used for the evaluation of a query. For example:

```
SELECT * FROM {ORDER} TAB3,TAB2,TAB1
WHERE …

SELECT * FROM TAB3 {INDEX INDEX1}
WHERE …
```

The first query will access the tables in the order they are specified in the from clause. The order directive can be placed after the keyword from in a subselect as well.

The second query will make use of index index1, if such an index or constraint is defined for tab3, when evaluating the query.

New Scalar Functions (V8.2.1)

Support for the scalar function bit_length, current_program and irand has been added.

The `bit_length` function returns the number of bits in a character or binary expression.

The current_program function returns the name of an entered program and null if no program is entered.

The irand function returns a random integer value and may take an optional parameter, which is used as a seed for a random sequence. Used in this way, as shown in the example, it is possible to generate the same random sequence on separate occasions. For example:

```
SET SEED = 32767;
SET INITIAL_VALUE = IRAND(SEED);
LOOP
SET RANDOM_VALUE = IRAND();
…
END LOOP;

SELECT BIT_LENGTH(C1) FROM TAB3
WHERE C1 like …
INSERT INTO EVENT_LOG values(coalesce(current_program(),current_user)),…
```

Online Backup (V8.2.1)

Support for online backup, has been added. With this new functionality it is possible to take a consistent backup while the system is accessible for read and write operations to other users.

All databanks, including the system databanks, may be backed up this way. A backup copy of the `LOGDB` databank, can be used as an incremental backup for all databanks in the system.

See the *Mimer SQL Reference Manual* for more details, especially the `START BACKUP`, `CREATE ONLINE BACKUP` and `COMMIT BACKUP` statements.

Mimer ESQL Embedded SQL

Support for Embedded Fortran (V9.2.4B)

Linux: The ESQL program now supports the -for option for embedded Fortran. The currently supported Fortran compiler is Intel (R) Fortran Compiler Version 8.

Support for long long Data Type in C (V9.2.1)

Embedded SQL/C now supports the C data type `long long`. This data type is an integer which always is 64 bits, regardless of architecture and platform.

Support for NCLOB struct in C (V9.2.1)

Embedded SQL/C now supports the C data type nclob, which is recognized by the ESQL/C preprocessor and converted to a struct. See *Programmer's Manual, Appendix A, Host Language Dependent Aspects*.

Support for NATIONAL CHARACTER LARGE OBJECT (NCLOB) type (V9.2.1)

Mimer ESQL now supports the SQL data type `NCLOB`. The client may use host variables of any character string data type. However non-Latin1 data may only be accessed using the wide character host variable data types `WCHAR_T` and `VARWCHAR_T`.

If an NCLOB containing non-Latin1 characters is fetched into a host variable of type CHARACTER, CHARACTER VARYING or CLOB, an error exception is raised.

WITH HOLD Cursors (V9.2.1)

From version 9.2 onwards, cursors may be kept open during transaction commits. This behavior is accomplished using the WITH HOLD clause within the cursor specification.

Examples:

```
exec sql DECLARE CRS CURSOR WITH HOLD FOR ... ;

exec sql ALLOCATE 'CRS' CURSOR WITH HOLD FOR ... ;
```

Block Fetching of Result Sets (V9.1.4)

ESQL applications will now, whenever possible, fetch result rows in blocks from the server. In effect this means that ESQL will, whenever the application want to fetch more data, get a number of rows from the server at once and store these in an internal buffer. Future fetches will read directly from the internal buffer until it is exhausted, when a new block of rows are requested from the server.

In most cases, this has a positive effect on performance, applications will communicate less with the server and thus improving its scalability. Communication overheads are also reduced. There are, however, a few cases when this might be detrimental to performance. One situation might be when one want the first result row as fast as possible, while there can take some time for the server to complete an entire block request. In these situations ESQL programmers may change the block fetch behavior with the session attribute FETCH SIZE. This attribute will provide a hint about a suitable fetch size, that is the number of rows to fetch in each block, to ESQL. ESQL, will whenever possible try to use the specified fetch size, but it may in practice use a fetch size smaller than specified. If the value is zero, the hint is ignored.

Example 1 (ensures that rows are transferred one at a time from the server):

```
EXEC SQL SET SESSION FETCH SIZE 1;
```

Example 2 (using a host variable):

```
EXEC SQL BEGIN DECLARE SECTION;
long fetch_size = 24;
EXEC SQL END DECLARE SECTION;
...
EXEC SQL SET SESSION FETCH SIZE :fetch_size;
```

Support for New SQL Statements (V8.2.1)

ESQL has full support for the new SQL statements introduced in Mimer SQL 8.2. For a description of the new syntax, see the *Mimer SQL Reference Manual.*

SQL Statement Classification (V8.2.1)

ESQL now writes a classification note after each pre-processed statement. For a description of the different classifications, see the *Mimer SQL Reference Manual*.

A flagger switch has also been introduced. A warning message will be returned for any SQL statement having a higher classification level than requested in the switch.

Full SQL92 Support (V8.2.1)

ESQL has full SQL92 support, including PSM (SQL/PSM-96) and triggers (SQL99). ESQL is thereby able to handle all the classification levels.

Character Pointer Support (V8.2.1)

ESQL now supports char-pointers and varchar-pointers in the DECLARE SECTION. For a description and examples, see the *Mimer SQL Programmer's Manual*.

Mimer ODBC – Open Database Connectivity

Indicator Variables Support SQL_DEFAULT_PARAM (V9.2.3)

Applications may now use the `SQL_DEFAULT_PARAM` value in indicator variables. The indicator variable is a word defined in the last parameter to `SQLBindParameter` or using the descriptor attribute `SQL_DESC_INDICATOR_PTR`. However, the SQL-2003 standard does not include the notion of a default value for parameters, and neither do Mimer SQL servers. The Mimer SQL server will therefore treat all default parameter values as being SQL NULL values.

Mimer Specific Keywords to SQLDriverConnect (V9.2.2)

To allow an application to connect without specifying a data source in the connection string, the following driver-specific keywords have been added for the Mimer ODBC Driver:

- `PROTOCOL`
- `NODE`
- `SERVICE`
- `INTERFACE`

See the ODBC chapter in *SQL Programmer's Manual* for a detailed description.

Connection Attribute SQL_ATTR_CONNECTION_DEAD (V9.2.1)

The connection attribute `SQL_ATTR_CONNECTION_DEAD` to poll connection status is now supported. By calling `SQLGetConnectAttr` with this attribute, the driver will return `SQL_CD_TRUE` if the connection has become unusable. `SQL_CD_FALSE` is returned if the driver believes the connection is alive and well. The driver won't try to diagnose the connection status, it merely returns what it has learned so far. Calling `SQLGetConnectAttr` with this attribute is therefore very cheap, but connection changes since the last network request will go by unnoticed.

Internal Array Fetching (V9.2.1)

The ODBC driver is now doing internal array fetches whenever possible. In most situations this will improve performance, but there may be some situations where it is important to get the first few rows returned to the application fast. In these situations, setting the row array size (`SQL_ATTR_ROW_ARRAY_SIZE` statement attribute) to anything, for instance one, will disable internal array fetching.

Mimer Specific Descriptor Fields (V9.2.1)

Mimer SQL supports large objects of up to 8 terabytes. This poses a problem to ODBC applications since the ODBC API specifies length fields to be 32 bits. An ODBC compliant application is therefore unable to work with any objects larger than 2 gigabytes. The SQL/CLI standard has the same problem, so we won't get much help there.

To remedy this situation, the Mimer ODBC driver has four vendor specific descriptor attributes. Each attribute mimics the behavior of an existing attribute, the only difference is that the existing attribute is working with a 32-bit integer while ours use 64-bit integers. These descriptor attributes are: SQL_DESC_DISPLAY_SIZE_64, SQL_DESC_LENGTH_64, SQL_DESC_OCTET_LENGTH_64, and SQL_DESC_OCTET_LENGTH_PTR_64.

See the ODBC chapter in *SQL Programmer's Manual* for a detailed description.

Support for NATIONAL CHARACTER LARGE OBJECT (NCLOB) type (V9.2.1)

ODBC now supports the SQL `NCLOB` type. The client may use any character string data type, but non-Latin1 data may only be accessed using the wide character client type `SQL_C_WCHAR`.

If an `NCLOB` containing non-Latin1 characters is fetched into a `SQL_C_CHAR` an error exception is raised.

Parameter Arrays in Conjunction With Data at Execution Parameters (V9.1.3)

Prior to version 9.1.3, Mimer ODBC did not support the usage of parameter arrays in conjunction with data at execution parameters. Data at execution parameters are specified by setting the word pointed to by `SQL_ATTR_OCTET_LENGTH_PTR` to `SQL_DATA_AT_EXEC`, or to a value defined by the `SQL_LEN_DATA_AT_EXEC(n)` macro. In these situations, the driver tells the application to supply parameter data by having the `SQLExecute` call returning `SQL_NEED_DATA`. In these situations the application should call `SQLParamData` to obtain the parameter requested by the driver. The data should then be supplied by calling the `SQLPutData` function. In Mimer ODBC version 9.1.3 and onwards, after the call to `SQLParamData`, the word pointed to by the statement attribute `SQL_ATTR_PARAM_PROCESSED_PTR` will contain information on which row the driver expects data to. The parameter is recognized by using the `SQLParamData` parameter `*ValutPtrPtr`, as before.

Unicode API (V9.1.1)

On Microsoft Windows systems, Mimer ODBC fully supports the wide character API used by the Microsoft ODBC Driver Manager. Whenever drivers export these functions (like `SQLConnectW`), the driver manager will use them with Unicode (`UTF-16`) arguments.

If the application has been compiled to use ASCII strings (`UNICODE=0 define`) the Driver Manager will convert the string to Unicode before calling the driver.

National Character Data Types (V9.1.1)

The `SQL_WCHAR` data type is now fully supported. Applications may now use the `SQL_WCHAR` data type to store and retrieve Unicode data.

Exactly which Unicode format is used depends on the platform. Microsoft Windows uses 16-bit characters while most other platforms use 32-bit characters.

Large OBject Support (V9.1.1)

The SQL data types `SQL_LONGVARBINARY` and `SQL_LONGVARCHAR` are now fully supported.

Each data type supports large character and binary objects of sizes up to 8 TB. 32-bit platforms have a smaller size limit of 2 GB.

ODBC Version 3 Support (V8.2.1)

Mimer ODBC driver for version 8.2 now supports ODBC 3.51.

New ODBC Data Types (V8.2.1)

The following SQL data types have been added:

- `SQL_BIGINT`
- `SQL_VARBINARY`
- `SQL_BINARY`
- `SQL_BIT`
- `SQL_GUID`

A binary column currently has a maximum length of 15 000 octets. The information about the maximum length of the data type is available in SQLGetTypeInfo.

A number of application program data types are now also supported:

- `SQL_C_BINARY`
- `SQL_C_INTERVAL`
- `SQL_C_NUMERIC`
- `SQL_C_SBIGINT`
- `SQL_C_UBIGINT`
- `SQL_C_GUID`

The preceding data types may be used freely in an ODBC application. SQL_C_BINARY on any other data type than SQL_BINARY will return the Mimer SQL internal representation of the data type and can only be used to store values in a column with the exact same data type.

New ODBC Data Types (V8.2.1)

The following SQL data types have been added:

- `SQL_BIT`
- `SQL_GUID`
- `SQL_TINYINT`

One new application program data type is now also supported:

- `SQL_C_BIT`

The preceding data types may be used freely in an ODBC application.

New ODBC Routines (V8.2.1)

Many new routines have been implemented. This enables an application to use ODBC 3 or ODBC 2 mode when accessing the Mimer database handler. The new routines are:

- `SQLAllocHandle`
- `SQLSetConnectAttr`
- `SQLGetConnectAttr`
- `SQLSetStmtAttr`
- `SQLGetStmtAttr`
- `SQLSetEnvAttr`
- `SQLGetEnvAttr`
- `SQLSetDescField`
- `SQLGetDescField`
- `SQLSetDescRec`
- `SQLGetDescRec`
- `SQLCopyDesc`
- `SQLColAttribute`
- `SQLFetchScroll`
- `SQLGetDiagField`
- `SQLGetDiagRec`
- `SQLCloseCursor`
- `SQLEndTran`
- `SQLFreeHandle`

Please refer to the ODBC documentation for specific information about the above routines.

ODBC Escape Clause Enhancements (V8.2.1)

A number of new escape clause constructions are now supported. These are:

- Intervals: `{INTERVAL …}`
- User defined functions: `{? = func(par1, par2,…)}`
- `Guid` escape clauses: `{guid '…'}`
- String functions:

  `{fn BIT_LENGTH …}`

  `{fn CHAR_LENGTH …}`

  `{fn CHARACTER_LENGTH ...}`

`{fn DIFFERENCE …}`

`{fn OCTET_LENGTH …}`

`{fn POSITION …}`

`{fn SOUNDEX …}`

- Date/time functions:

  `{fn CURRENT_DATE …}`

  `{fn CURRENT_TIME …}`

  `{fn CURRENT_TIMESTAMP …}`

  `{fn EXTRACT …}`

Row Wise Parameter Binding (V8.2.1)

Row-wise parameter binding is now available through the statement attribute `SQL_ATTR_PARAM_BIND_TYPE`. Previously, this was only supported for fetch operations.

New Statement Options (V8.2.1)

The statement option `SQL_QUERY_TIMEOUT` is now supported.

Mimer BSQL

SET EXECUTE ON/OFF Statement (V9.2.2)

When `EXECUTE` is set to `OFF`, statements will be compiled but not executed. This can be useful when checking statements for errors or when using the explain facility.

SET EXPLAIN ON/OFF Statement (V9.2.2)

When `EXPLAIN` is active, a query plan will be shown. This plan shows in which order tables are accessed and if any indexes are used when executing the query.

Example:

```
SQL>set explain on;
SQL>set execute off;
SQL>select * from orders as o, order_items as oi
SQL&  where o.order_id = 12
SQL&  and o.order_id = oi.order_id;

L1:
Sequential read MIMER_STORE.ORDERS , end of table goto end
compare, no hit goto L1
L2:
Sequential read MIMER_STORE.ORDER_ITEMS , end of table goto L1
compare, no hit goto L2
Record found, resume at L2
end:
```

See *Mimer SQL User's Manual* for more details.

Additional Parameters for BSQL Command (V9.2.1)

When starting BSQL it is now possible to supply username and password to the command to facilitate an automatic login.

Example VMS:

```
BSQL /USERNAME="MIMER_STORE" /PASSWORD="GoodiesRUs"
```

Example Unix:

```
bsql -u MIMER_STORE -p GoodiesRUs
```

It is also possible to give a command as argument in which case BSQL executes the query and then quits.

Example:

```
bsql -q"select * from music_store" -uMUSIC_STORE -pGoodiesRUs
```

This command can be any valid SQL/BSQL statement.

List/Describe Functions for Collations (V9.2.1)

It is possible to get information about collations. This can be obtained by using the menus or commands directly.

Example:

```
SQL> describe collation information_schema.albanian_1;
```

UTIL Functionality (V9.2.1)

Some of the functionality formerly found in UTIL has been moved to BSQL. This includes the readlog functionality and logic for recreating system databanks at startup.

The menu for readlog is invoked by using the statement `READLOG`, see *Mimer SQL User Manual*.

New Data Type NCLOB (V9.2.1)

The new data type `NATIONAL CHARACTER LARGE OBJECT` (i.e. `NCLOB`) can be used in BSQL.

Enhanced File I/O (V9.1.1)

Mimer BSQL can now handle files in `UTF8`, `UTF16`, `UTF16BE`, `UTF16LE`, `UTF32`, `UTF32BE` and `UTF32LE` formats. For example:

```
LOG OUTPUT ON 'sales' AS UTF16;
```

For more information, see the `LOG` and `READ` commands in the *Mimer SQL User's Manual*.

Limiting the Amount of Data Displayed (V9.1.1)

Two new BSQL commands have been introduced. These are:

```
SET MAX_CHARACTER_LENGTH value

SET MAX_BINARY_LENGTH value
```

The keywords `MAX_CHARACTER_LENGTH` and `MAX_BINARY_LENGTH` can be abbreviated to `MC` and `MB` respectively.

Example:

```
SET MC 20;
SELECT * FROM INFORMATION_SCHEMA.COLUMNS;
```

In the example above, only the first 20 characters of all (character type) columns will be displayed.

Support for New SQL Syntax (V8.2.1)

BSQL has full support for the new SQL functionality introduced in Mimer SQL 8.2. For a complete description of how to use BSQL, see the *Mimer SQL User's Manual*.

Delimited Data in LOAD/UNLOAD (V8.2.1)

BSQL now supports the capability to load and unload data delimited by a specific character rather than the fixed formats previously supported.

Example:

```
LOAD FROM 'ACCOUNT_DATA' INTO ACCOUNT DELIMITER ';';
```

DbVisualizer

DbVisualizer (V9.2.4D)

The SQL tool DbVisualizer is bundled with Mimer SQL.

Mimer Utilities

Updated DBC Syntax (V9.2.1)

The DBC program has a new, optional parameter for sysdb filename. This filename is required if any tables are using collations. If not specified, the correct sort order for such tables is unknown.

DBOPEN Command Line Arguments (V9.2.1)

The `DBOPEN` program now accepts username and passwords as command line arguments.

Example (Unix)

```
dbopen -u SYSADM -p GoddiesRUs server1
```

Example (VMS)

```
dbopen /username=SYSADM /password=GoodiesRUs server1
```

EXPTOLOAD (V9.2.1)

Since `UTIL` has been removed from version 9.2 of Mimer SQL, an Exptoload program is available for conversion of old `UTIL` export files to files that can be loaded by the `LOAD/UNLOAD` functionality.

Mimer UTIL

New Functions

The following sections document new functions.

128 Character Object Name Length (V8.2.1)

The Export/Import utility now supports object names with up to 128 characters.

Binary Data Type Support (V8.2.1)

The Export/Import utility now supports the data types `BINARY` and `BINARY VARYING`.

Load/Unload with User Defined Delimiter (V8.2.1)

The Export/Import utility now supports load/unload with a user-defined delimiter.

VMS Specific Features

The TCPCONTROL Command Procedure (V9.2.1)

The `TCPCONTROL` command procedure can now be used to start, stop and check the `MIMTCP` processes. Please see the *Mimer SQL Open VMS Guide* for further information.

UNIX Specific Features

DbVisualizer (V9.2.4F)

The DbVisualizer SQL tool is now bundled with Mimer SQL. For details on DbVisualizer, please see http://www.dbvis.com/products/dbvis/.

DbVisualizer must be started from the File Manager or from the Application Manager in the desktop. If using the File Manager, /usr/local/bin/dbvis-mimer can be opened. In the Application Manager “DbVisualizer (Mimer-SQL)” is found under the Applications tab.

.dumper.sh Script (V9.2.1)

When dbinstall is executed, a .dumper.sh script is generated in the database server home directory. This script is defined as the DumpScript parameter in the multidefs file, which will cause it to be run if the system fails. The .dumper.sh script executes some commands to verify system resource allocation.

New Method For Local Communication (V9.1.4)

The “local” communication method has been redesigned. This communication method is used by Mimer SQL clients that are on the same node as the database server. Previous versions used a pipe to send control messages between the client and the server. The new version will mainly use a System V semaphore to perform the communication; when a client or a thread in the server needs to wait for each other, they can do so by waiting for a semaphore to be set. By using semaphores rather than pipes, applications that communicate much (i.e. don't use array fetches or procedures) can get a significant performance boost.

Whenever a client connects to the database server, the server creates a new System V semaphore set to be used for communication purposes. Since most Unix systems have a limit on the number of such sets that can be created, this operation can fail. If this happens, and error message will be returned to the client, and a message will also be written in the mimer.log file. The OS error message will contain the text:

```
semget: [ENOSPC] No space left on device
```

(This message will only be displayed on the client side if the application is programmed to look up error messages and display them fully.)

The command `ipcs` can be used to list all semaphore sets in the system. The semaphore sets created by a Mimer SQL server will have a key of `0x00000000` or `0xffffffff` (depending on Unix platform), access mode of `0600` and have 2 semaphores in the set. The semaphore set is created by the user that is running the dbserver process and is owned by the user that runs the client process.

The database server is responsible for removing the semaphores it allocates. Semaphores are removed whenever:

- A client disconnects normally.
- When the server detects that a client has disappeared without disconnecting.
- When the server is shut down.
- If the server should crash it will remove all used semaphores after it has created a dump directory.
- When a server is started, it will remove any old semaphores.
- Whenever the command `mimcontrol -k` is given.

Note that some Unix systems have system limits that control the number of semaphores that can be created. These limits include:

`SEMMNI`	Maximum number of semaphore sets. One semaphore set is needed per local connection
`SEMMNS`	Maximum number of semaphores. Should be twice the value of `SEMMNI` for Mimer SQL purposes.
`SEMMSL`	Maximum number of semaphores per semid. Mimer SQL uses two semaphores in each semaphore set. For Mimer SQL usage, `SEMMSL` must be at least 2.

Please consult the manuals for your operating system for more information about how these limits are used.

Large Files Support (V8.2.4)

Linux: Large files are now supported on Linux.

Automatic Startup (V8.2.2)

Automatic database server start and stop functionality is provided. The new utility programs `mimautoset` and `mimservers` are used for this purpose.

Man Pages (V8.2.2)

UNIX man pages for Mimer SQL are distributed and installed when installing the server. On some UNIX versions, the `-F` option is required when using the `man` command to read the man pages.

RPM Support (V8.2.2)

Linux: RPM as an installation tool is supported on Linux platforms.

JDBC Driver (V8.2.2)

The Mimer JDBC driver is included in the distribution. The driver is found in the `lib` directory and is named `mimjdbc-x_x.jar`. An example program, `example.java`, is included in the examples directory.

There will be further development of the JDBC driver which will not require a new server release. New versions of the JDBC driver will be made available on the Mimer SQL developer site.

New Utilities – Version 8.2 (V8.2.2)

The following table shows all new utility programs that are supplied with Mimer SQL version 8.2.

Utility	Function
`dbfiles`	Lists the databank files for a server, as stored in the data dictionary.
`mimautoset`	Switches on/off the automatic server start and stop functionality or gives the current state.
`mimdbfiles`	Lists the databank file names for a server, as stored in the UNIX file system. Can also be used to change the ownership of the databank files. The new owner should be the one that is dedicated to manage the database server.
`mimhome`	Displays home directory for the effective user.
`mimlicense`	Administrates the license keys.
`mimowner`	Displays name of user that is dedicated to manage a specific server.
`mimservers`	Starts/stops all version 8.2 dbservers defined in /etc/sqlhosts.

I/O Threads (V8.1.3)

Linux:	To support parallel I/O execution on Linux, specific threads performing I/O requests are implemented. The number of I/O threads used by a database server is configurable from the multidefs file, where an additional parameter `IOThreads` is available for Linux.

POSIX Threads (V8.1.1)

Mimer SQL version 8 is based upon POSIX threads. When using threads, only one process is seen for an executing database server.

Linux:	Currently on Linux, each thread is given a unique process ID that can be seen in a process status report.

New Utilities – Version 8.1 (V8.1.1)

The following table shows all new utility programs that are supplied with Mimer SQL version 8.1.

Program	Short explanation	Used by
`asctoexpc`	Program used for export file conversion. Replaces the old mimchop program.	
`dbinstall`	Command used to install a new database.	
`dbserver`	The database server program. You start dbserver using the mimcontrol.	`mimcontrol`
`exptoasc`	Program used for export file conversion. Replaces the old mimunchop program.	
`mimadmin`	Menu based database server administration utility.	
`mimcontrol`	Program to manage database servers.	`mimadmin`
`mimdevenv`	Command used to create a beginner's development environment.	`dbinstall`
`mimexampldb`	Command used to create an example database environment.	`dbinstall`
`mimhosts`	Program to manage the `/etc/sqlhosts` file. See *Managing /etc/sqlhosts Using mimhosts (V8.1.1)* on page *24* for more information.	`dbinstall`, `mimadmin`
`miminfo`	Program to monitor database servers. Replaces the old mimserv program.	`mimadmin`

Program	Short explanation	Used by
`mimlink`	Command used to link Mimer SQL libraries and executables to /usr/lib and /usr/bin, respectively. See *Symbolic Links to Mimer (V8.1.1)* on page *25* for more information.	`dbinstall`
`mimlistdb`	Command used to list started database servers.	`mimadmin`
`mimpath`	Command used to get the path to databank locations.	`mimadmin`, `mimlink`, `mimunlink`
`mimsdbgen`	Command used to create initial system databanks (starts the sdbgen program with default parameters).	`dbinstall`
`mimstatln`	Command used to follow a symbolic link.	`dbinstall`, `mimlink`, `mimunlink`
`mimtcp`	Program to manage TCP port dispatching, i.e. distributing incoming connect-attempts to the requested database server. See *Managing Client Connects Using mimtcp (V8.1.1)* on page *25* for more information.	
`mimuninstall`	Command to uninstall Mimer SQL version 8.	
`mimunlink`	Command used to remove symbolic links from /usr/bin and /usr/lib, previously created by the mimlink command.	`mimuninstall`
`mimversion`	Command used to get the installed Mimer SQL version.	`mimadmin`, `mimlink`, `mimunlink`

Managing /etc/sqlhosts Using mimhosts (V8.1.1)

The database server registration file, `/etc/sqlhosts`, can be administered by using the new administration utility called `mimhosts`. The utility can do the following tasks:

- Automatically create the `/etc/sqlhosts` file if it does not exist
- Display the complete `/etc/sqlhosts` file
- Add or delete an entry for a local or remote database
- List registered local or remote databases
- List or change the default database

The `mimadmin` program presents a menu based user interface to the `mimhosts` program.

The `mimhosts` program is automatically invoked when the `dbinstall` program is used to add a database.

Managing Client Connects Using mimtcp (V8.1.1)

The `mimtcp` program is a TCP/IP port number dispatcher that recognizes Mimer SQL client requests. It can only act as a dispatcher for Mimer SQL version 8 database servers. The program listens to the TCP/IP port number 1360 that is assigned for Mimer SQL use. It reads the handshake message and forwards the connection to the desired database server on the node.

The use of the `mimtcp` program is encouraged and the default installation enables the feature. This means that the `/etc/inetd.conf` file is updated to automatically start the `mimtcp` program when a connection request is received on the Mimer port.

Symbolic Links to Mimer (V8.1.1)

When installing Mimer SQL, symbolic links are created that link Mimer SQL executables, libraries and man pages to `/usr/bin`, `/usr/lib` and `/usr/man`, respectively.

The symbolic links are created by using the command `mimlink` (invoked by the `miminstall` command). A corresponding `mimunlink` command is available to remove the links.

Windows Specific Features

Home Directory Path (V9.2.5)

The home directory for a database may now contain a list of directories separated with semicolons. When the server searches for an unqualified filename (a name with drive and directory specification), the directories in the path are tried in order to find the databank file. When a new file with an unqualified filename is created it is created in the first directory in the path.

Service Descriptions (V9.2.5)

On Windows XP systems the Mimer SQL Database and network services now includes a description that is shown in the Services panel of the Control Panel.

Mimer ODBC Trace (V9.2.2)

Mimer ODBC Trace is a tool for getting the details out of an ODBC session. It can for example be used for debugging, monitoring and analyzing purposes. For each ODBC statement the tool can trace:

- the elapsed time in server
- the SQL statement used (if applicable)
- the ODBC function used
- the timestamp for the ODBC call
- the ODBC handle used

- the calling application process ID and name
- the return code for the ODBC call

The software consists of two parts, the Mimer ODBC Trace Administration tool and the Mimer ODBC Trace DLL (that substitutes the standard Microsoft ODBC Trace DLL).

The Mimer ODBC Trace Administration tool has two main functions:

- Configuration - to setup prerequisites for the trace logging and to start/stop logging
- Presentation - to select and display the records from a logging file.

The Mimer ODBC Trace DLL is activated from the administration tool.

See the *Getting Started on Windows* guide for a more complete description of Mimer ODBC Trace.

Note: Before using the Mimer ODBC Trace tool, please see the following notes:

- When changing any settings using the administration tool, you should restart any application for which you want the changes to be used. The reason for this is that the settings will only be read when the DLL is loaded by the application, and thus the DLL has to be reloaded for any changes to take effect.
- The administration tool changes the registry entries in `HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI\ODBC`. If there is any problem look there in order to see if your changes are applied.
- The Windows default ODBC Administrator may also change those registry entries. Currently there is a problem concerning the cooperating use between these two administration tools. Watch out which DLL is selected, as both administrators may start or stop tracing with anyone of the DLL’s.

Automatic Database Server Memory Configuration (V9.2.1)

When a new local database definition is made in the Mimer Administrator, the size of the database server cache, the bufferpool, is now determined by examining the amount of memory present in a machine configuration. There are three different page sizes in the cache: 2K, 16K, and 64K. The 2K buffers are assigned 12.5%, 16K buffers 8.33%, and 64K buffer 5% of the available memory.

Getting Started after Reboot (V8.2.4)

If a reboot is needed for a first-time install, the Getting Started guide and Mimer Administrator may be activated by a dialog box displayed during the reboot sequence.

Automatic Server Restart (V8.2.1)

The database server can now restart itself automatically after an internal failure. This option is enabled by default, but may be turned off in the Local database configuration dialog's Server tab in the Mimer Administrator.

New File Extensions (V8.2.1)

Two new file extensions (`mdmp` and `mcfg`) are registered when the software is installed.

`Mcfg` is used for license keys and other Mimer Administrator related configurations.

`Mdmp` is used for dump files that can be examined using the Mimer Info utility.

New Routines for ODBC Driver Configuration (V8.2.1)

Support has been added for the ODBC routines `SQLConfigDataSource` and `SQLConfigDriver`. See the *Mimer SQL Packaging Guide* for further information.

Mac OS X Specific Features

The Mac OS X Documentation (V9.2.4)

Currently, if nothing else is stated, the Mimer SQL documentation should be read and used as if the Mac OS X platform is equal to Unix/Linux.

The applications Mimer SQL database Install and Mimer SQL database Admin have both built-in help libraries.

DbVisualizer (V9.2.4D)

The SQL tool DbVisualizer is bundled with Mimer SQL.

Mimer SQL Mobile Specific Features

Mimer SQL Engine Support on Pocket PC (V9.2.5)

The complete Mimer SQL Engine database server is now supported on Pocket PC. This means that the entire range of SQL commands are supported on Pocket PC including ALTER, CREATE and so on. The server type can be specified in the Server tab for local database definitions. Note that you may have to minimize the keyboard on the device to see this setting in the dialog.

Pocket PC Memory (V9.2.5)

It is now possible to use larger bufferpools in the Pocket PC environment. This was previously limited to 12 MB, but is now, instead, limited by the amount of memory available.

Databank Size Reduction (V9.2.2H)

When packaging Mimer databanks for deployment, unnecessary system views, collations and system tables are now removed. The views eligible for removal are the views in the INFORMATION_SCHEMA and MIMER schemas). Note that the views are only removed if they are not accessed or required by the statements created before deployment. Thus, this operation resembles the obfuscation phase MIDP Java programmers frequently execute in the sense that unused code is omitted from the deployed product. Similarly, collations not used by the application schema are also removed.

The views in the FIPS_DOCUMENTATION and INFO_SCHEM schemas are deprecated and they are never accessible on the mobile device.

Import Database From Handheld Device (V9.2.1)

With the new version it is possible to import an entire database and all related files from the handheld device or the emulation environment. Right-click on the local database you want to import to in the Mimer Administrator to start the Database Import Wizard.

New Server Type on Desktop (V9.2.1)

The database server on the desktop can now emulate a Mimer SQL database server. By using the mobile server you can easily verify that your applications work well with a Mimer SQL database server.

You can select server type in the database administration dialog for local databases under the tab “Server”. Please note, that the server must be restarted before the setting becomes active.

Mobile Server on Desktop (V9.2.1)

To make it easier to test a Mimer SQL database configuration it is now possible to start a Mobile server on the PC. This is done by setting server type to Mobile in the Server tab of a local database configuration. This is further described in the Mimer Administrator help.

Chapter 2

Changed Features and Functions

This chapter describes changed features and functionality introduced into Mimer SQL in previous versions.

The Mimer SQL Database Server

Smaller Page Sizes Used at End of LOB Storage (V9.2.5)

The strategy for storing large objects (LOB's) in pages is changed, so the use of disk space is made more economic.

Example:

Previously if a LOB required between 2K and up to 16K, one 16K page was always allocated. Now if the LOB is between 2K and 12K, instead so many 2K pages as required are allocated.

Also the writing of LOB pages is now more efficient as only the occupied part of a page is written to disk.

Compressed Commit Set (V9.2.5)

The commit set is used to secure transaction data at commit. This data is now stored using compression which can decrease the space needed in TRANSDB. It may also allow for more cases where group commit may be used (i.e. several transactions can be secured on disk in a single I/O operation).

More Efficient Bitmap Handling (V9.2.5)

The system no longer has to write bitmap pages to disk when a databank file size has changed. This allows the system to operate even more efficiently when dynamic allocation of file space is needed. In addition the internal algorithm for finding free pages within the bitmaps has been improved.

New Compressed Form for Data Storage (V9.2.5)

A new compress algorithm is now used. This new algorithm is more efficient especially for records containing Unicode data (NATIONAL CHARACTER data types in Mimer).

Tables using the new algorithm can be seen by using the DBC program, which displays those tables with the information “Mimer LZU compression”.

Performance Enhancements (V8.2.4)

A number of significant performance enhancements have been implemented in version 8.2. Some performance enhancements are described in the ODBC chapter, as these are specific for ODBC clients.

Faster Access with Long Read-only Transactions (V8.2.4)

When the transaction cache grows large due to long transactions the response time in the system decreases somewhat.

In the new version, the system can access the transaction cache much faster if the long running transactions are read-only transactions. Therefore, it is important to, whenever possible, set transactions read-only when they are expected to run over a longer period of time. This will improve performance for both the read-only transaction, as well as other concurrent transactions.

Long running transactions that are read/write are not recommended and should be split into several smaller transactions. This will also minimize the impact if the transaction is aborted and must be re-executed.

Index Lookup Only (V8.2.1)

When accessing secondary indexes in Mimer SQL the system has previously always performed a lookup in the base table as well. From version 8.2 the system will only look in the secondary index table when the following conditions are true:

- The query only references columns in the index. An index contains the columns defined as the index and the primary key (unless the table does not have a primary key).

  It may therefore be beneficial to add columns to an index if this allows the optimizer to skip the base table. The cost is, of course, increased storage usage and an additional overhead in secondary index handling.
- The index is consistent. An index may be inconsistent if the base table is: stored in a NULL databank; or the index has been converted from an earlier version of Mimer SQL and UPDATE STATISTICS has not been run; or if the system databank TRANSDB is recreated. An index can be made consistent by updating the statistics for the table on which the index is defined.

The cost based optimizer in Mimer SQL has been updated to take into consideration the lowered cost of accessing secondary indexes.

New Reorganization Algorithm (V8.2.1)

The algorithm for reorganizing b*-trees has been revised. The change allows the system to accumulate several reorganizations before actually writing the changes to disk. The new algorithm still maintains the properties for careful replacement, i.e. the database is always consistent both on disk and in memory.

The number of I/Os done by the system is significantly lower than before. This is particularly evident for tables that make many changes to a small part of the b*-tree.

For tables located in NULL databanks, this change results in the table being flushed to disk less often. That is, if the machine crashes, the number of changes that are lost may now be greater than before. For tables under transaction control (i.e. TRANS or LOG databank), the change has no impact except the system works faster than before.

Bitmap Lookup (V8.2.1)

For databanks with many bitmaps (files larger than 32 Mb), the system will remember which bitmap to allocate pages in. Previously a sequential scan of the bitmaps was performed.

The system is also more efficient when finding free bits within the bitmap.

Databank Check in Background (V8.2.1)

After an improper shutdown of the system, a consistency check of all databank files is automatically performed when the system is restarted. This operation has now been moved to the background threads. For large databanks, the restart time is significantly reduced.

While the background check is being performed, small local checks are made when accessing tables in the databank. Thus, there is no danger in accessing the database even though the background check is not complete.

If the database server is shut down while a check is ongoing, the check is aborted and subsequently restarted the next time the databank is accessed.

Block Pre-fetch (V8.2.1)

Whenever the system is going to do a sequential scan of all or part of a table it will request pre-fetch. Pre-fetch allows read I/Os to be performed before a block is actually needed. When the block is later referenced it may either be in memory already, or the amount of time to wait for the read to complete will be reduced.

Caching of Compiled Queries (V8.2.1)

In version 8.1, the compiled procedure or SQL statement was released as soon as there were no references to it. In version 8.2 the system will cache compiled procedures and SQL statements. This will allow for faster database access, particularly in systems with few users and heavy use of dynamic SQL (ODBC for example).

Optimized Sort/Load Operations (V8.2.1)

A number of improvements have been made in the sort/load/merge steps. Among the improvements are: larger block sizes, use of pre-fetch, and more parallel merge steps where each merge is allowed more pages than before. This affects many operations that internally use the load facility.

Enhanced Work Table Management (V8.2.1)

Performance for queries, using a temporary table for storing intermediate results, has in many cases, been improved. In earlier versions, an index tree was built for the temporary table. This is unnecessary in most cases, as the data only needs to be accessed sequentially.

Optimized Aggregate Functions Expressions (V8.2.1)

The performance of expressions such as `select max(column) + 1` has been improved.

Group by Optimization (V8.2.1)

Group by queries now uses the new worktable mechanism described in 2.2.1.9, if the columns in the group by clause are retrieved in sorting order.

New Licensing System (V8.2.1)

The licensing system is now called Mimer SQL License Key. The old term MRS is no longer used.

The major changes are as follows:

- The licensing system in version 8.2 is completely server based. That is, no keys are required for the database clients.
- Several database servers on the same machine can share the available number of concurrent users defined by the key.
- Several keys may be combined. For example, if two applications with Mimer SQL bundled are installed on a machine, the keys provided by both applications form a combined key used by the system.
- The system can handle licenses for applications distributed with Mimer SQL.
- The format of the key is less strict than before, and may contain comments, blanks and new lines.

Mimer SQL Structured Query Language

Using Tables With LOB Columns in Triggers (V9.2.4)

In earlier versions of Mimer SQL it was not allowed to define a trigger on a table with a column defined with a lob type. This has now been changed. For example it is possible to write triggers to log all operations on a table, e.g.

```
CREATE TABLE master     (pnr VARCHAR(10),
                         image BLOB,
                         PRIMARY KEY(pnr));

CREATE TABLE masterCopy (byWhom VARCHAR(128),
                         action VARCHAR(10),
                         pnr VARCHAR(10),
                         image BLOB,
                         whatTime TIMESTAMP,
                         PRIMARY KEY(byWhom, action, whatTime));

CREATE TRIGGER copyMaster AFTER INSERT ON master
REFERENCING NEW TABLE AS n
BEGIN ATOMIC
    INSERT INTO masterCopy
        SELECT SESSION_USER, 'INSERT', pnr, image, LOCALTIMESTAMP FROM n;
END
```

The only restriction remaining is that it is not possible to refer to `LOB` columns in the `NEW TABLE` in an `INSTEAD OF` trigger.

Collations and NCHAR (V9.2.1)

Collations now support the national character data types.

New Columns in INFORMATION_SCHEMA Views (V9.2.1)

The view `INFORMATION_SCHEMA.EXT_IDENTS` has an additional column `IDENT_SCHEMA`.

The view `INFORMATION_SCHEMA.EXT_STATEMENTS` has two additional columns, `IS_SCROLLABLE` and `IS_FORWARD_ONLY`.

See *Mimer SQL Reference Manual, Chapter 13* for more information on these views.

Changed Privilege Check in Views (V9.2.1)

The views `INFORMATION_SCHEMA.TABLE_CONSTRAINTS` and `INFORMATION_SCHEMA.KEY_COLUMN_USAGE` now show information on tables where the current user has some privilege other than select. Earlier it only showed information if the current user was the creator of the table.

The view `INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS` now shows data if the current user has some privilege, other than select, on the referenced table.

Cascade/Restrict (V8.2.1)

The default for `CASCADE` and `RESTRICT` for all `DROP` and `REVOKE` commands has been changed. Previously, the default was `CASCADE`. This has now been changed to `RESTRICT`.

The reason for this change is that the dependencies are more complex, now that procedures, functions and triggers reference objects.

When `CASCADE` is in effect, any procedure, function, and/or trigger is automatically dropped. By changing the default, the risk of inadvertently removing objects that should be kept is reduced.

Privilege Changes (V8.2.1)

Previously, the right to perform certain commands was controlled by EXECUTE privilege on three system defined PROGRAM idents. These programs have now been replaced by system privileges:

Old PROGRAM ident	New privilege
`MIMER_BR`	`BACKUP`
`MIMER_SC`	`STATISTICS`
`MIMER_SW`	`SHADOW`

The command to grant or revoke the privilege has changed accordingly:

Old command	New command

`grant EXECUTE on MIMER_BR to ident`
`revoke EXECUTE on MIMER_BR from ident`
	`grant BACKUP to ident`
	`revoke BACKUP from ident`

Old command	New command

`grant EXECUTE on MIMER_SC to ident`
`revoke EXECUTE on MIMER_SC from ident`
	`grant STATISTICS to ident`
	`revoke STATISTICS from ident`

`grant EXECUTE on MIMER_SW to ident`
`revoke EXECUTE on MIMER_SW from ident`
	`grant SHADOW to ident`
	`revoke SHADOW from ident`

No Duplicate Privileges (V8.2.1)

The system no longer maintains duplicate privileges granted from one ident to another. The algorithm for recursively revoking privileges has, because of this, been simplified.

If grant option for a privilege is revoked from ident A, and A no longer has grant option from any other source, then the privilege is recursively revoked from other idents who have been granted the privilege by A.

The old algorithm considered the set of privileges A had at the time the privilege was granted (rather than the current privileges).

Update Statistics Cleans Indexes (V8.2.1)

Secondary indexes in databanks with option `TRANS` or `LOG` are always kept consistent with the base table.

However, in a `NULL` databank it may be possible for a secondary index to contain more rows than the base table. When a databank changes from `NULL` to `TRANS` or `LOG`, the indexes are still marked as inconsistent. Indexes may also be inconsistent when upgrading from earlier versions of Mimer SQL.

By running an `UPDATE STATISTICS` command the indexes will be made consistent. This can be done concurrently with any other activity in the system.

When the index is consistent the SQL optimizer does not have to visit the base table when the index contains all the requested columns. The speedup is significant when this occurs (more than twice as fast!).

To find out if an index is inconsistent or not, use the following select statement:

```
SELECT OBJECT_SCHEMA,OBJECT_NAME,IS_CONSISTENT
FROM SYSTEM.OBJECTS, SYSTEM.TABLE_CONSTRAINTS
WHERE OBJECT_TYPE = 'INDEX' AND OBJECT_SYSID = CONSTRAINT_SYSID;
```

This will show the consistency for all indexes. The `OBJECT_SCHEMA` column contains the schema name for the index and the `OBJECT_NAME` column contains the index name. The column `IS_CONSISTENT` has the value '`YES`' when the index is consistent. The select statement can normally only be executed by the system administrator (`SYSADM`). Other users can execute the statement if granted `SELECT` access to the involved tables.

Representation of DOUBLE PRECISION and FLOAT (V8.2.1)

Values declared as `DOUBLE PRECISION` or `FLOAT` need to be able to represent one additional bit. Therefore, the data types use more space than before.

CREATE IDENT Changes (V8.2.1)

When creating a user or program ident, the system will automatically create a schema for the user. Consequently, the user is allowed to create domains, synonyms, sequences, procedures, functions and views in the schema. For the user to be able to create other objects, appropriate privileges are needed. This retains compatibility with earlier versions.

It is also possible to create an ident without a schema. This ident may not, without additional grants, create any type of objects. For example:

```
CREATE IDENT ADAM AS USER USING `ADAMPASSWORD`
WITHOUT SCHEMA;
```

CREATE DATABANK Changes (V8.2.1)

When creating a databank, it is no longer required to specify initial size, file name and databank option. For example:

```
CREATE DATABANK D;

CREATE DATABANK D WITH LOG OPTION;
```

The default initial databank size is 1000 pages, the default file name is the same as the databank name, and the default databank option is `TRANS`.

Hex-constants (V8.2.1)

A hexadecimal constant (e.g. `x'204C6172732042657267'`) is typed as a bit string and not a character literal. A bit string is not comparable nor assignment compatible with a character value, without an explicit cast expression.

A hexadecimal constant can be used in conjunction with binary columns without explicit conversion.

DDL-statements in Transactions (V8.2.1)

In earlier versions of Mimer SQL, a Data Definition statement was not allowed if a transaction was started. This has now been changed so DDL statements may be rolled-back or committed as other statements.

There are some restrictions on the use of DDL statements, though. In this version only one DDL statement is allowed in a transaction. If a transaction is started and a DML statement has been executed, the execution of a DDL statement will cause an error. If a DDL-statement has been executed within a still active transaction and a new statement is executed, the previous DDL-statement is committed.

In the future it will be possible to have multiple DDL statements in one transaction. Thus, giving an explicit commit statement after each DDL-statement will be forward compatible.

Mimer ODBC – Open Database Connectivity

SQLBrowseConnect and SQLDriverConnect (V9.2.2)

The ODBC Mimer Driver behavior for the `DSN`, `DRIVER` and `DATABASE` keywords found in a connection string has been changed.

Previously, when `DATABASE` was specified, the database name was always used for lookup even though a `DSN` was specified first. If only `DSN` was specified, the data source name was first tried to be found as a data source and if not found a second try was made by using it as a database name.

Now, if the `DSN` and `DRIVER` keywords are included in the same connection string, the driver use whichever keyword appears first (and the other is ignored). If `DRIVER` is used then the `DATABASE` keyword is also required.

Mimer Product Name (V9.2.1)

The Mimer product name has changed from ‘MIMER/DB’ to ‘Mimer SQL’. This reflects error messages as well as some information items returned by `SQLGetInfo`, such as `SQL_DBMS_NAME` and `SQL_DBMS_VER`.

Performance Enhancements (V8.2.1)

The performance enhancements described are specific for ODBC. A number of general performance improvements are also described in the Database Server chapter.

Less Server Communication with ODBC (V8.2.1)

The version 8.2 database server now handles auto-commit. Previously, the client had to perform the auto-commit, resulting in one extra communication for each auto-commit.

In addition, when using `SQLExecDirect` without any host variables, the server immediately executes the request without an additional server communication.

Example:

```
SQLExecDirect(hStmt,
   "INSERT INTO TAB1(C1,C2) VALUES ('A', 22)", SQL_NTS);
```

The above example required 3 communications (compile, execute and auto-commit) in version 8.1 and requires only 1 communication in version 8.2.

Automatic Read-only Cursors with Auto-commit (V8.2.1)

The database server is especially fast when it knows that no write operations will be performed. In this case, no read set will be constructed (which keeps track of all rows seen by the application).

Previously, this could only be done when the connection attribute `SQL_ATTR_ACCESS_MODE` was set to `SQL_MODE_READ_ONLY`. When auto-commit is enabled, the database server knows that the only statement in the transaction will be the SELECT. Therefore, it can, temporarily, set the connection read-only. This gives the same effect as if `SQL_MODE_READ_ONLY` had been specified.

SELECT FOR UPDATE (V8.2.1)

If a `SELECT FOR UPDATE` statement is executed while the value of `SQL_ATTR_CONCURRENCY` is set to `SQL_CONCUR_READ_ONLY`, an error will now be returned.

Note: `SQL_CONCUR_READ_ONLY` is the default value. This behavior is in accordance with ODBC 3.

SQLSTATE Changes (V8.2.1)

The driver works in two modes depending on if the application expects ODBC 2 SQLSTATEs or ODBC 3 values. ODBC 2 SQLSTATEs are used when either `SQLAllocEnv` is called, or `SQLSetEnvAttr` is called with `SQL_ATTR_ODBC_VERSION` set to `SQL_OV_ODBC2`.

Changes to ODBC Routines (V8.2.1)

- `SQLFetch` now supports block cursors according to ODBC 3.
- `SQLGetInfo`. All information items, according to ODBC 3, are supported by the Mimer ODBC driver. All in all, 129 information items are supported.
- `SQLGetFunctions` now returns all new routines. Support is also added for a new mode where information about all supported routines is returned in one call.
- SQLColumns returns a result set with the following columns added: `COLUMN_DEF`, `SQL_DATA_TYPE`, `SQL_DATETIME_SUB`, `CHAR_OCTET_LENGTH`, `ORDINAL_POSITION`, `IS_NULLABLE`.
- `SQLGetTypeInfo` returns a result set with the following columns added: `SQL_DATATYPE`, `SQL_DATETIME_SUB`.
- `SQLPrimaryKeys` returns a result set with the following column added: `DEFERRABILITY`.
- `SQLProcedureColumns` returns a result set with the following columns added: `COLUMN_DEF`, `SQL_DATA_TYPE`.

All catalog functions support the `SQL_ATTR_METADATA_ID` attribute.

Truncate Without Warning by SQLFetch (V8.2.1)

`SQLFetch` will no longer return a warning when blanks are truncated from a character value or nulls are truncated from a binary value.

Mimer SQL Data Type FLOAT (V8.2.1)

The Mimer SQL data type FLOAT with precision is no longer associated with different ODBC SQL data types depending on precision.

It is now always associated with `SQL_FLOAT`.

Extended SQLGetData Functionality (V8.2.1)

`SQLGetData` now supports block cursors and also allows getting unbound columns in any order.

Use `SQLGetInfo(…, SQL_GETDATA_EXTENSIONS, …)` to obtain information on extended `SQLGetData` support.

Mimer BSQL

The CONNECT Statement (V8.2.4)

The BSQL `CONNECT` statement will now use any database name supplied when starting BSQL.

For example:

```
$bsql lokdb

BSQL>connect;
User: LOKADM
Password: secret
```

This will connect to the database `lokdb`, whereas in earlier versions the connection would be attempted with the default database.

The `CONNECT TO` default and `CONNECT TO ''` ... statements are not affected.

List and Describe (V8.2.1)

List and describe functions for all objects introduced in version 8.2, have been added.

Example:

```
LIST SEQUENCES;
DESCRIBE FUNCTION TANGO.SQRT;
```

Show Settings (V8.2.1)

The show settings command now includes information about transaction isolation level and transaction read write mode.

Mimer Utilities

MIMLOAD (V9.1.1)

A new program, `MIMLOAD`, for loading definitions and data to and from databases has been implemented. For more information, see the *Mimer SQL System Management Handbook*.

Mimer UTIL

Load/Unload Menu (V8.2.1)

The Load/Unload menu in the Export/Import utility has been updated. Now, it contains two additional options where load/unload can be made with a user-defined delimiter.

These new menu options had to be added in the existing context of the menu and some other options had to be moved to new menu positions. Thus, the old menu options for unload now have other numbers.

Using UTIL in a Script (V8.2.1)

If the UTIL program fails, it will return a return-code to the calling environment. Therefore, it is possible to test for failures when using UTIL in scripts by using the standard mechanisms in the script language.

Using INFORMATION_SCHEMA (V8.2.1)

Previously, the Export/Import utility gathered information about objects to export by using the MIMER-views. In Mimer SQL version 8.2, standard `INFORMATION_SCHEMA` dictionary views are supported. This gives slightly different access to objects and their relations.

As the `INFORMATION_SCHEMA` views are more restrictive, compared to the MIMER system views, it is only possible to export objects that are created by the currently connected ident.

VMS Specific Features

Dump Files and MIMER.LOG (V8.2.4)

When a server creates dump files in a dump directory, it will also create a directory entry for the `MIMER.LOG` file if the dump directory is on the same disk as the `MIMER.LOG` file.

MIMINFO Command Syntax for Selecting a Database (V8.2.4)

In previous versions, the `MIMINFO` command had a `/DATABASE` switch that had to be used to select a database. For example:

```
$ MIMINFO/DATABASE=ORDER/USERS
```

The command syntax has been changed so that the database is specified as a command parameter. The `/DATABASE` switch has been removed. This makes the syntax more like the other Mimer commands such as `MIMCONTROL`.

The previous example becomes:

```
$ MIMINFO/USERS ORDER
```

MIMTCP Process (V8.2.2)

In Mimer SQL version 7, a network server (`NETSRVM`) was installed so that it started when connection requests were received on the `MIMER` port (usually port number 1360).

In Mimer SQL version 8, these installation steps are not required. A `MIMTCP` process is started for each TCP/IP port that a Mimer SQL database is listening to. This process will accept new connections from TCP/IP clients and hand over the connections to the appropriate database server. This allows several database servers to share a TCP/IP port number.

Installation is Pre-linked (V8.1.1)

Before version 8 of Mimer SQL, the executable files were linked in the installation procedure by running the `MIMBUILD` command procedure.

In version 8, the installation is completely built and is ready to use after installing the directory tree. The MIMBUILD command procedure is removed.

The MIMBUILD configuration file (CONFIG.DAT) is also removed. All configuration parameters have been moved to the MULTIDEFS and SINGLEDEFS parameter files (see the *Mimer SQL VMS Guide*).

No Single-user Mode Libraries (V8.1.1)

In previous versions of Mimer SQL, applications (and Mimer SQL programs such as BSQL) could be linked in both single-user and multi-user mode. An application linked in single-user mode would access all LOCAL databases directly, and multi-user mode programs would access those databases by connecting to a database server.

In Mimer SQL version 8, there is only one Mimer SQL database library to link to. Normally, all applications execute as if they were linked in multi-user mode.

However, by defining the logical name MIMER_MODE to SINGLE, the single-user mode library (MIMLIB8:MIMDBS_xxx.EXE) will be loaded dynamically. Thus, all Mimer SQL programs have the capacity to run in both single-user and multi-user mode depending on how the logical name MIMER_MODE is set.

The S or M suffix is dropped from the executable programs found in MIMEXE8 so that programs such as BSQLM or UTILS now have the names BSQL and UTIL.

New SINGLEDEFS.DAT Parameter (V8.1.1)

In previous versions of Mimer SQL, some single-user mode database parameters such as the size of the bufferpool was controlled in the CONFIG.DAT file.

In Mimer SQL version 8, all parameters for single-user systems are found in the SINGLEDEFS.DAT file, which must be located in the home directory of the database (the directory where the SYSDB8 file is found). If the file is absent, the single-user mode library will use default values for all parameters.

An example file for SINGLEDEFS.DAT containing the default values used by the system can be found in the example directory (MIMEXAMPLES8). Always make a copy of this file to a database home directory and change the copy.

Changed Directory Structure (V8.1.1)

The directory structure has changed slightly from the ‘run-time tree’ found in older Mimer SQL versions. (The distribution tree ([MIMAXP7]) is no longer distributed.) A new sub-directory ([.EXAMPLES]) contains all code examples that are distributed.

The name of the top directory contains the version number of the Mimer SQL distribution. This simplifies future upgrades since those distributions will have new unique names. By using the MIMSETUP8 command procedure, it is easy to switch between Mimer SQL versions. Old Mimer SQL versions can be deleted by simply removing the distribution tree. (No Mimer SQL files are placed on SYS$SHARE.)

The directory structure is fully explained in the *Mimer SQL VMS Guide*.

Multi-vendor TCP/IP Support (V8.1.1)

In Mimer SQL version 8, all TCP/IP access is performed by using $QIO primitives to the BG device driver. Most TCP/IP vendors for VMS (including Digital UCX, TCPware and Multinet) support this device driver. No special actions have to be performed during the Mimer SQL installation to support these vendors.

Changed MIMSETUP8 Parameters (V8.1.1)

The `MIMSETUP8` command procedure defines logical names and installs shared images so that a specific Mimer SQL version can be used. The command procedure is similar to the old `MIMSETUP7` procedure, but the parameter specifying the run-time tree has been removed. This information is retrieved by the `MIMSETUP8` command procedure by examining the location of the `MIMSETUP8.COM` file. (The `MIMSETUP8` procedure will not work if the file is moved away from the distribution tree.)

The `MIMSETUP8` procedure can now also remove defined logical names and de-install images.

More information about the `MIMSETUP8` procedure is found in the *Mimer SQL VMS Guide*.

DECNET Object Name is the Database Name (V8.1.1)

In Mimer SQL version 7, a network object named `MIMER` had to be declared, by using the `NCP` command, to allow clients to connect to a database server using DECNET. When a connection request was received, the DECNET software started a process that executed the `NETSRVM` program. `NETSRVM` could then connect to the desired database.

In Mimer SQL version 8, all DECNET clients connects directly to the database server. When the database server starts, it automatically listens for new DECNET connections, so no manual DECNET installation is necessary. However, each database server started on a machine needs to define a unique network object that it can listen on. The object name chosen is the same name as the database that the server operates on.

If the DECNET client is running Mimer SQL version 7, the default service name (object name) for DECNET was `MIMER`. If a version 7 installation is to connect to a version 8 server, the service name should be changed in `SQLHOSTS.DAT` to be the same as the database name.

If the DECNET client is running Mimer SQL version 8, the default service name for DECNET is the same name as the database name. If a version 8 client is to connect to a version 7 server using DECNET, the service name should be changed to `MIMER`.

No Files on SYS$SHARE (V8.1.1)

Mimer SQL version 7 placed a number of files on the system directories. These included files for shareable images, and section files for mapped shared data areas.

Mimer SQL version 8 does not place any file in the system directories. Shareable image files reside in the library directory (`MIMLIB8`). All shared memory resources use the page files as backing storage.

MULTIDEFS.DAT (V8.1.1)

The contents of the `MULTIDEFS.DAT` file has changed since Mimer SQL version 7. It is recommended that the contents of this file is revised when upgrading to Mimer SQL version 8.

All parameters in `MULTIDEFS.DAT` now have a default value. The default value for a parameter is used when the parameter is absent from the file. This means that it is possible (but not necessarily desirable) to have a completely empty `MULTIDEFS.DAT` file.

The `MULTIDEFS.DAT` file is described in the *Mimer SQL VMS Guide*.

UNIX Specific Features

dbinstall Execution (V9.2.1)

The dbinstall program is now changed so that it can be executed towards an already existing database. If used this way, missing objects for the database can be recreated. For example, if the .dumper.sh script is missing, it will be recreated.

New I/O Primitives Used on Solaris (V9.2.1)

Solaris: Earlier the Solaris Asynchronous I/O package was used, but since there were problems detected (especially when using third party disk management products) the I/O system has been replaced. Now there are dedicated I/O-threads in the Mimer SQL configuration that perform all I/O.

The DumpScript Parameter in the multidefs File (V8.2.4F)

Using the `DumpScript` parameter, you can specify a command that will execute if a database server crashes. This command will be executed in a separate process (the database server uses the system() call).

The following string substitutions will be performed before the command is used:

`%p` – The `pid` of the aborting database server process

`%%` – Insert a single `%` character

We recommend that you don't change the value of this parameter, unless you are directed to do so by Mimer SQL support staff.

Automatic Server Dump Facility (V8.2.4)

The database server’s dump facility which automatically creates a dump directory containing information useful for Mimer SQL support personnel, now creates dumps when the `SIGBUS` signal is encountered.

Updated Error Tracing (V8.2.4)

If the Mimer SQL database server is aborted due to a signal reception, detailed information about the cause of the interruption is written to the database server log file, i.e. `mimer.log`. This information is mainly for helping Mimer SQL support personnel in tracing possible errors.

By using the `multidefs` parameter `DumpScript`, see *The DumpScript Parameter in the multidefs File (V8.2.4F)* on page *42*, additional information can be gathered from an aborted database server.

Once all dump files are generated, the server will attempt to create a core file. Note that the core file will only be created if the database server process has the appropriate process limits.

You should set the process limits for the process that starts the database server (runs the `mimcontrol -s` command) since the process limits are inherited by the database server process.

Linux: On Linux, you can check what process limits you have by using the command:

```
$ ulimit -a
```

To set a limit for the core files use the -c switch, for example:

```
$ ulimit -c 999999
```

If you set the limit to zero (0), no core file will be generated.

Root Not Required (V8.2.2)

When creating a database by executing the `dbinstall` command, any user, not only root, can be defined to manage the `dbserver`.

Example makefile Version 8.2 (V8.2.2)

The example `makefile` no longer needs the `MIMER_HOME` environment variable to be defined.

Licensing System (V8.2.2)

The license system has been changed. All software keys are stored in one file: `/etc/mimerkey`. This file is administrated using the `mimlicense` utility.

SDBGEN (V8.2.2)

Previous versions of the `SDBGEN` program always created the database in the current directory. In this version of `SDBGEN`, it is possible to supply a database name which will be used when looking up the directory in which the database files should be created.

The database name can be supplied as a parameter to the `SDBGEN` program. If no parameter is given the value of `MIMER_DATABASE` is used and if this variable is undefined the default database in `/etc/sqlhosts` is used.

The `SDBGEN` program is also used to upgrade the database from earlier versions.

The mimsdbgen program is no longer included in the distribution as it is possible to supply parameters to the SDBGEN program.

For more information on SDBGEN, see the *Mimer SQL System Management Handbook*.

Default Number of Users (V8.2.2)

The default license key included in the distribution allows 10 simultaneous users for test and development. This license key is installed when the system is installed and is subject to the license agreement.

tmp Not Used (V8.2.2)

The /tmp directory is no longer used for storage of temporary files. Instead, the directory defined by the environment variable TMPDIR is used for such storage. If this variable is not set, the directory .mimer_tmp is created in the users home directory for such storage.

For log file storage, the directory .mimer_log is created in the user's home directory.

Client Channel Limit (V8.1.3D)

Previously, there was a hard limit (50) on the number of client channels, i.e. connections, a client process could do.

Now this limit is adjustable up to 2000 by setting the environment variable MIMER_MAXCLIENTCHAN. If not set, the default is 50. The environment variable assignment should be made for the client process accessing the database server.

When the limit is reached, an error message like the following will be displayed:

```
SQL>CONNECT TO 'CUSTOMERS' AS 'C1' USER 'SYSADM' USING 'MUDDY';
MIMER/DB fatal error -21024 in function CONNECT
No available channel id number
```

Asynchronous I/O Limit (V8.1.3D)

Previously, there was a hard limit (128) on the number of parallel asynchronous I/Os that could be started. On some systems, situations could arise where system limits or system resources could not cope with a specific amount of parallel I/O operations. Error messages like the following one could be displayed:

```
aio_write: [EAGAIN] Resource temporarily unavailable
```

Now this limit is adjustable up to 128 by setting the environment variable MIMER_MAXASYNC. If not set, the default is 40. The environment variable assignment should be made for the user process starting the database server, i.e. root.

If this limit is set too low it may, for example, not be possible to start a database server.

Error messages like the one in the following example session may be displayed if the limit is set too low in comparison to the server configuration:

```
# export MIMER_MAXASYNC=8
# mimcontrol -s customers
MIMER/DB Startup Error
         Error when initializing bufferpool (DKSTA9)
         Too many kernel and shadow servers specified: 5+2, max = 5

2000-05-04 16:28:12.26   <Error>
Database server not operational

#
```

MemLock Default in multidefs (V8.1.3D)

Previously, the default value for the `MemLock` parameter in the database server `multidefs` file was 1, i.e. enabled. This is now changed to 0 (disabled).

(The default value is generated when the `multidefs` file is created, i.e. when the database server is started and no `multidefs` file exists).

Menu System, mimadmin (V8.1.3)

The `mimadmin` interface is updated so that more information is showed by default:

```
Target database server: customers
Database server state: Stopped
Database server home: /d1/customers
MIMER installation used: /opt/mimer813A
```

mimunlink (V8.1.3)

The `mimunlink` command is updated so that it now can remove links for an installation that is not of the same version as the `mimunlink` command itself. This can be useful in situations where installations are moved or removed without first removing these links to `/usr/lib` and `/usr/bin`.

If the versions differ, there is no guarantee that everything is unlinked since `mimunlink` only knows about links for the set of files included in the current version.

MIMER_HOME Not Required (V8.1.3)

Previously in version 8 the `MIMER_HOME` environment variable was required to be set to point out the Mimer SQL installation used. If not set, various error situation could arise.

Single-user Shared Library Lookup

In the following case, the single-user shared library lookup failed:

```
# bsql -s
MIMER/DB fatal error -21040 in function CONNECT
Could not map library for single-user mode, OS Error message:
'dlopen: ./lib/libmimdbs.so: cannot open shared object file: No such file'
#
```

Now the common operating system shared library lookup is used.

That is, when linking the Mimer SQL shared libraries to `/usr/lib`, they are automatically located by the runtime loader.

If the libraries are not linked to `/usr/lib` the environment variable `LD_LIBRARY_PATH` (or corresponding) should be set to point out the library location.

HP-UX: For HP-UX, the `MIMER_HOME` environment variable setting is still required to locate the single-user shared library.

Database Server Startup

The following example illustrates how a database server startup failed if `MIMER_HOME` was not defined correctly:

```
# mimcontrol -s customers
1999-10-12 10:46:39.77   <Error>
The environment variable MIMER_HOME must point to the MIMER distribution
#
```

Now, the path used to find the `mimcontrol` program is also used to find the `dbserver` program.

Example makefile Version 8.1 (V8.1.3)

The example `makefile` is updated, mainly for easier usage when creating ODBC applications.

Note: The example `makefile` still needs the `MIMER_HOME` environment variable to be defined (machine specific compiler and loader options are included from the `makeopt` file found in the examples directory of the Mimer SQL installation).

Database Server Alert Messages (V8.1.2)

In earlier versions, the `MULTIADM` user received an e-mail when fatal errors occurred. In Mimer SQL version 8, the UNIX `syslog` function is used which means that if a fatal error occurs, a message is printed on the console and the same message is written in a system log file.

The `Oper` parameter in the `multidefs` parameter file can be used if you want to send an e-mail in fatal error situations. If `Oper` is defined, an e-mail containing the error message will be sent using the string assigned to `Oper` as the recipient(s).

Administration Environment (V8.1.1)

In Mimer SQL version 8, the administration environment is simplified. The database server is now installed, executed and administered by the superuser, i.e. root. Previously, the users `MIMERADM` and `MULTIADM` were created for managing the installed software and the database server(s). These users are no longer needed.

The Mimer SQL software is ready to use when installed. That is, shared libraries, programs, data files, etc. do not need any specific management or profiling.

The databases can be managed by using the `mimadmin` command. This command presents a menu based system that invokes underlying standalone programs, such as `mimcontrol`, `mimhosts`, `miminfo`, etc.

No Single-user Mode Programs (V8.1.1)

In previous versions of Mimer SQL, applications (and Mimer programs such as `bsql`) could be linked in single-user mode and multi-user mode. An application linked in single-user mode would access all LOCAL databases directly and multi-user mode programs would access those databases by connecting to a database server.

In Mimer SQL version 8, there is only one Mimer SQL database library to link to. Normally, all applications execute as if they were linked in multi-user mode. However, by defining the environment variable `MIMER_MODE` to be `SINGLE`, the single-user mode library will be loaded dynamically.

All Mimer SQL programs have the capacity to run in both single-user mode and multi-user mode by using command line switches or by defining the environment variable `MIMER_MODE`.

The `s` or `m` suffix is dropped from the executable programs installed. Programs such as `bsqlm` or `utils` now have the names `bsql` and `util` and only exist in one form.

Changed Directory Structure (V8.1.1)

The directory structure has changed slightly from the one found in older Mimer SQL versions.

When installing Mimer SQL version 8, the `miminstall` command will prompt for the installation directory. A sub-directory named to reflect the actual Mimer SQL version, e.g. `mimer811A`, will be created relative to the installation directory. (This naming convention simplifies future upgrades since those distributions will have new unique names).

Beneath this top directory the following sub-directories can be found:

`bin` for executable files.

`doc` for documentation, e.g. PDF-files and README files.

`examples` for various examples.

`lib` for libraries.

`man` for man pages

DB Server Configuration File, multidefs (V8.1.1)

The `.multidefs` file that existed for database servers in Mimer SQL version 7, is now renamed to `multidefs` (without a leading dot). Most of the parameters contained in it have also changed since Mimer SQL version 7.

The `multidefs` file is automatically created for each database server if it does not exist at startup. Default values are specified for all the parameters. For detailed information on the multidefs file see the *Mimer SQL System Management Handbook*.

Note: Old configuration files, i.e. from Mimer SQL version 7, should not be used in Mimer SQL version 8.

Database Registry File, /etc/sqlhosts (V8.1.1)

When a database is installed using the `dbinstall` command, the database will be automatically registered in the `/etc/sqlhosts` file.

If the `sqlhosts` file does not exist, it will be created with a default layout which is different to the sample `sqlhosts` file distributed in earlier Mimer SQL versions. If an existing Mimer SQL site wishes to use the new style `sqlhosts` file, simply rename the old one, execute the `mimhosts` command and use it to add the relevant databases to the new `sqlhosts` file which were defined in the old `sqlhosts` file.

If the `dbinstall` (or `mimhosts`) command is executed when an `/etc/sqlhosts` file from a previous Mimer SQL version exists, the new database entry will be added correctly and all existing entries in the file will be reformatted to fit the new `sqlhosts` database entry layout. Comments are left as is.

For more information on the `sqlhosts` file see the *Mimer SQL System Management Handbook*.

TCP Ports for Database Servers (V8.1.1)

In Mimer SQL version 7, a network server (`netsrvm`) was installed so that it started when connection requests were received on the Mimer port number (usually port 1360).

In Mimer SQL version 8, these installation steps are not needed. Basically, database clients connect directly to the database server, using a unique TCP/IP port, which implies that each server on a node must be set up to listen to a different port.

However, in the default installation, the `mimtcp` program is used which allows clients to always establish a connection to the port number 1360, no matter which Mimer SQL version 8 database server is targeted.

The parameter `TCPPort` in the `multidefs` file (exists for each database server) determines the port number that the database server listens to. However, the default is `inetd` which indicates that the server uses the `mimtcp` program for client connects.

If the `mimtcp` program is not used, a TCP/IP port number should be specified for the `TCPPort` parameter. If several database servers are started on the same machine, a unique TCP/IP port number has to be allocated for each. This means that the correct port number must be specified in the `/etc/sqlhosts` file on machine of each client wishing to connect to a particular database server. If a connection request for a specific database is made to the wrong database server, the server will refuse the connection.

Database Server Aliases (V8.1.1)

In Mimer SQL version 7, a database could be given several names by adding several entries to the `/etc/sqlhosts` file that pointed to the same database.

In Mimer SQL version 8 this is no longer possible. Each database should be given a name that is unique to avoid possible confusion.

Database Server Dump Directory (V8.1.1)

As in earlier versions, the server placed dump files in a dump directory if it crashed. The name of this directory is changed and contains the current month, day and time, in the format `MonDD_HHMM`. The dump directory will normally be placed under the home directory of the database (where the `sysdb8.dbf` file resides). The location of the dump directory can be changed by using the `DumpPath` parameter in the `multidefs` parameter file.

ODBC Support (V8.1.1)

Mimer includes an ODBC driver, but in the Mimer SQL version 8 distribution there is no ODBC Driver Manager provided. In Mimer SQL version 7, the runtime part of the Visigenic ODBC Driver Manager was bundled.

Theoretically, any ODBC Driver Manager can be used.

Windows Specific Features

Mimer Administrator (V8.2.1)

The Mimer Administrator now provides more information in the overview window. Details about the different objects are now displayed. The main window has been made re-sizable.

Local database run-status is indicated by the color of the icon. Green means the database server is running, yellow means logins are disabled or shutting down, red means the database server is stopped, and finally, gray means status is unknown.

It is possible to control a local database using a popup menu. Activate the menu by right-clicking on the database name.

The Mimer Administrator has been enhanced to handle several concurrent license keys. This support is available under a separate tab in the Mimer Administrator's main window. The contents of an added key can be examined by double-clicking the corresponding entry.

It is also possible to view the contents of a key in the dialog in order to update a key.

Mimer Info Enhancements (V8.2.1)

The Mimer Info utility can now display information from the dump directories that are created when the database server malfunctions.

It is also possible to select a new database without restarting the utility.

DB-check Enhancements (V8.2.1)

The Windows based Databank (DB) Check utility has been enhanced so that it is now possible to view the output and navigate the output using the arrow keys while the DB-check is in progress. In addition, an ongoing check may now be cancelled.

The open file dialog allows several files to be selected.

Mimer SQL Mobile Specific Features

Filenames Altered During Export (V9.2.5)

When databank files are exported to a PDA the system now removes any drive and directory specifications from the filenames in the data dictionary. This allows the new home directory path features (see *Home Directory Path (V9.2.5)* on page *25*) to be used on the device. I.e. databank files can be placed in different locations such as memory sticks and so on.

Mimer Explorer Enhancements (V9.2.1)

Mimer Explorer now displays information whether a statement is available as SCROLL, NO SCROLL, or both. This feature is only available when using a server of version 9.2 or later.

Chapter 3

Corrected Features and Functions

This chapter describes corrected features and functionality introduced into Mimer SQL in previous versions.

The Mimer SQL Database Server

Execution Interrupted by Scheduler (V9.2.5)

A problem has been corrected which sometimes could cause incomplete statements to terminate with an "Execution interrupted by scheduler" error.

The problem would only occur for expensive statements that are both prepared and executed with the same client request (such as SQLExecDirect for ODBC clients) on a fully occupied database server.

Online Backup Improvements (V9.2.4)

When an online backup was made the transaction cache was not released properly. This could cause degradation of performance for long online backup sequences. This has now been corrected.

Occasionally the online backup would produce errors such as "Table identifier invalid" or "Internal inconsistency detected". This was caused by a concurrent copying of TRANSDB content while applying changes to other backup databanks. This has now been corrected.

A problem has also been corrected for databanks containing BLOB or CLOB data. The problem occasionally caused an inconsistent backup databank where some of the LOB data references were corrupt. The inconsistency can be detected by using the DBCHECK utility on the backup databank.

Recreating a Smaller LOGDB (V9.2.4)

A problem has been corrected regarding recreation of LOGDB. The problem could occur under rare circumstances if the recreated LOGDB file was smaller than the preceding file. The problem would cause I/O errors to be logged when dropping the log next time or when restarting after a crash.

Improved Index Handling (V9.2.4)

The SQL compiler's index handling has been improved, as corrections have been done to the query optimizer.

ORDER BY With Many Indexes (V9.2.2G)

`SELECT` with `ORDER BY` had problems when too many indexes were involved. This is now corrected.

More Than 32 Foreign Key References From a Table (V9.2.2)

If a table was defined with more than 32 foreign key references the error message "Internal inconsistency detected" was signalled at insert or update.

Interrupted UPDATE STATISTICS (V9.2.2)

If a databank was offline, or if a table was not accessible for any reason, updating statistics was interrupted and no statistic was gathered for the following tables. This is now changed, so those tables are ignored. A warning message would have been nice, but is not introduced.

Implicit Databank Verification Causing Loop (V9.2.2)

If a databank has more than one rootpage (i.e. holding many tables) and there are tables with primary key columns using collation definition, the implicit databank verification could cause an infinite loop. Implicit databank verification will only be run if the database server is not correctly stopped.

Using Columns With Collations in Primary Keys or Indexes (V9.2.1C)

There is a problem in version 9.2.1A and 9.2.1B which can cause a database server to hang when accessing a table where a primary key or index is sorted using a non-default collation.

This problem is now corrected.

XA Transactions – Migrating Between Processes (V9.2.1)

When distributed transactions were used with the XA protocol, it was possible for a transaction to be handled by one process and then subsequently handled by another process. When this was done, the ownership for the reference counts for underlying objects used by the transaction was not properly migrated to the other process. The effect could be that exclusive operations incorrectly were allowed or, in rare circumstances, the objects actually were closed before the transaction completes. This is now corrected.

XA Transactions – Object Reference After Restart (V9.2.1)

When a machine crashes, any distributed transactions that were in the prepared state (and not yet committed), are activated upon restart of the server.

However, reference counts for the objects used by the transaction were not set properly. The effect could be that exclusive operations may be incorrectly allowed or, in rare circumstances, the objects actually were closed before the transaction completed. This is now corrected.

Database Hang During Databank Verification (V9.2.1)

The database server would previously hang during the internal verification, when the last page of the databank file was a bitmap page. This occurs rarely as only every 16256th block is a bitmap and the extension of the databank file occurs in chunks.

Next Values of Sequences (V8.2.4)

In some cases, the next_value of sequence_name function would return previously used sequence numbers.

This problem has now been corrected.

Online Backup Problem (V8.2.4E)

If the system ran out of transaction state table entries during an online backup which included TRANSDB, the server would display an error message and terminate. This occurred when one of the background threads accessed the TRANSDB backup incorrectly.

This has now been corrected.

Background Thread Loop (V8.2.4E)

If a reorganization of the commit set in TRANSDB occurred simultaneously with commit of transactions with more than 16K of data changes, the background threads would, in rare circumstances (due to timing conditions) enter an infinite loop. This has now been corrected.

Multiple Updates of Same Record with Secondary Index Access (V8.2.4E)

If: the same row was updated more than once, and the index row was read from the index base table, and the table row from the transaction cache, and the statement was subsequently rolled back due to some other error, an inconsistency was found on the write and an error log entry with the text `Write set corrupt` was written.

The write set is now handled properly in this case.

Invalid Timestamp Messages During online Backup (V8.2.4E)

During an online backup which included LOGDB, error messages saying there was an invalid timestamp, were sometimes written to the mimer.log file. The error message was incorrect, and the backup was properly made.

The code that produced the incorrect error log message has been corrected.

Creating Temporary Tables Using MIMER/PG (V8.2.4E)

There was a problem with version 8 servers which, in some situations, could prevent MIMER/PG from creating temporary tables.

This problem has now been corrected.

Many Request Threads and High Load (V8.2.4)

In earlier versions of 8.2, the Mimer SQL database server could crash due to a problem with its scheduler. The problem could occur when every request thread was occupied with a large request and the server was configured for use of more than 10 request threads. This problem is now corrected.

Growing T-cache (V8.2.4)

In previous versions, the transaction cache grew when inserting or updating rows that were not accessed any more, for example, when inserting rows in ascending primary key order. This caused SQLDB to grow.

In this latest version, the system detects when there are no active transactions at all in the system and, whenever this is the case, drops the entire transaction cache. This makes transaction cache handling more efficient than before.

Further improvements have been made when long read-only transactions are active in the system. In this case, the transaction cache grows but all newly started transactions only access small parts of the transaction cache. Therefore, it is important to make sure that transactions are made read-only as often as possible This is now done automatically in the BSQL program.

Error Handling and Low Memory (V8.2.4)

The error handling in the database server in low memory situations has been improved.

In previous versions, connections could be dropped when the server memory pool was low.

DDL Statements and Older Clients (V8.2.4)

Previously, when using a version 7.2 client, such as BSQL, with a version 8.2. database server, all DDL statements would fail. This has now been corrected.

Mimer ESQL Clients (V8.2.4)

In earlier versions of 8.2, a Mimer ESQL client using SQL statements with more than 180 result columns and input parameters could cause the Mimer SQL database server to crash when several users were active on the server. This problem is now corrected.

Commit Set Problems (V8.2.4)

When only very large transactions were made (larger than 3 megabytes), the commit set continued to grow, causing TRANSDB databank to grow. This has now been corrected.

In rare circumstances, the system reported a corrupt commit set. A subsequent restart of the server cleared this problem. The cause of this problem has now been corrected.

Accessing Views in Read Only Transactions (V8.2.3)

Previously, there was a problem when accessing views in read only transactions if the view had to be recompiled. For instance, a view has to be recompiled after using the UPDATE STATISTICS statement. Previously, this caused an infinite loop. This problem has been fixed.

Mimer SQL Structured Query Language

Date Arithmetics (V9.2.5)

When calculating an interval between two dates, days within a month was not considered. As an example

```
SELECT (DATE '2005-04-23' - DATE '2004-04-24') YEAR
FROM information_schema.ext_onerow;
```

will serve. In previous versions the result would be one year but this is now corrected in that the query will return the result zero years.

UPDATE WHERE CURRENT and Primary Key Duplicates (V9.2.4)

An UPDATE WHERE CURRENT that updates the primary key was previously allowing the updated record to replace an existing record with the new primary key value. The corrected behavior is to return a primary key duplicate error.

Compound SQL Statements and Large Objects (V9.2.4)

It is now allowed to use LOB columns in compound SQL.

Example:

```
DELETE FROM lobtab WHERE c1 = 2; INSERT INTO lobtab VALUES (99,?);
```

LOB Values Incorrectly Returned as NULL (V9.2.3B)

There has been a problem where some inserted `BLOB`, `CLOB` or `NCLOB` values are treated and returned as NULL values by the database server.

The problem can occur on systems where the database server has been started and stopped many times.

This problem is now corrected and values that previously were returned incorrectly as NULL will now be returned correctly.

LIKE Predicate in Functions or Procedures (V9.2.2H)

A problem has been corrected regarding use of the `LIKE` predicate in functions or procedures.

The problem could cause a “Compiled LIKE pattern corrupt” error message or possibly terminate the database server. The problem would only occur when several users access a function or procedure where the character pattern in the `LIKE` predicate is specified using a variable.

CAST From Timestamp to Time (V9.2.2G)

Using the following code:

```
DECLARE ftime TIMESTAMP;
DECLARE ntime TIME;

SET ftime = LOCALTIMESTAMP;
SET ntime = CAST(ftime AS TIME);
```

in a PSM routine caused an unjustified internal error. This has now been corrected.

Defining Foreign Keys With On Delete (V9.2.2G)

When defining a foreign key constraint with an `ON DELETE` action where the added length of the column names used in the constraint exceeded approximately 150 characters, the statement would fail or make it impossible to access the table at a later stage.

This has now been corrected.

Numeric Collations (V9.2.2G)

The `[numeric on]` option did not always work correctly in a collation, that contained a contraction. Swedish Å, Ä, and Ö are examples of contractions. `A` + digits and `O` + digits did not always sort as expected. This is now corrected.

MAX/MIN on Variable Length Columns (V9.2.2E)

`MAX` and `MIN` did in some cases have problem with the length of variable length data types. This is now corrected.

MAX/MIN on NCHAR Column Included in Index (V9.2.2E)

Using the `MAX` or `MIN` functions on a column of `NCHAR` or `NVARCHAR` type, where the column is included in any type of index (primary key, unique constraint, etc.), raised the internal error “Invalid OPCODE”. This problem is now corrected.

Varchar Function in SELECT (V9.2.2E)

If a function returning variable length data (`VARCHAR`, `VARBINARY` or `NVARCHAR`) was invoked from a `SELECT`, `UPDATE` or `DELETE` statement, the length value was not handled correctly. This problem is now corrected.

UPDATE With LOCALTIMESTAMP and NULL (V9.2.2E)

Depending on the column order, an `UPDATE` with both `LOCALTIMESTAMP` and `NULL` could in some cases raise an “constraint violation” error. This has now been corrected.

Error When Updating Foreign Keys (V9.2.2E)

There has been a problem where an update of a table could stop the database server if the table references or is referenced by a table that is inaccessible.

This problem is now corrected.

Multiple Column Indexes (V9.2.2E)

There has been a problem in previous 9.2 versions where an update of a table with an multiple column index could result in an “Internal inconsistency” error.

This problem is now corrected.

UPDATE With Subquery (V9.2.2E)

A problem with `UPDATE`, containing a correlated scalar subquery in a `SET` clause, has been corrected.

Temporary Storage Areas in PSM (V9.2.2E)

PSM has some problems with handling of temporary storage areas used for intermediate results when evaluating complex expressions. This can cause the inappropriate error “Length error or incorrect value found during data type conversion”.

This has now been corrected.

Case Statement in Continue Handler (V9.2.2E)

Using case statements within a exception handler could cause the server to crash. This has now been corrected.

Multi Statement SQL With Triggers (V9.2.2E)

Using a multi statement, e.g.

```
DELETE FROM orders; INSERT INTO orders SELECT * FROM ind;
```

when there are triggers defined on the `IND` table could cause unpredictable behavior. This has now been corrected.

Missing Cascade Effects (V9.2.2E)

The cascade effects of `DROP SCHEMA`/`DROP TABLE` did not drop all multiple triggers with references to the dropped table. This has now been corrected.

Scrollable Cursor and WCHAR Host Variables (V9.2.2D)

A scrollable cursor for a `SELECT` statement with host variables for input values (in the `WHERE` clause), where those variables were used to provide variable length data, raised the error “Negative overflow for VARCHAR length”. This has now been corrected.

LEFT JOIN and Descending Index (V9.2.2D)

`LEFT JOIN` in combination with a descending index read data from the index table instead of the base table. As a result of this, random data was returned. This has now been corrected.

Declare Cursor WHERE CURRENT Mismatch in PSM (V9.2.2C)

In some cases when declaring a cursor with a subselect and then using that cursor in an `UPDATE WHERE CURRENT` could erroneously cause the error the table used in update where current is not the same as in the cursor declaration.

This has now been corrected.

Invalid Argument Allowed for ASCII_CODE in PSM (V9.2.2C)

When using the function `ASCII_CODE` in PSM it accepted nchar data as argument (with an invalid result). Such an attempt will now cause a semantical error saying that only character data can be used as arguments to the `ASCII_CODE` function. To get the code point for nchar data, the `UNICODE_CODE` function should be used.

Table Lookup Problem in PSM (V9.2.2C)

Using a statement such as

```
INSERT INTO t1(c1,c2) SELECT * FROM v1
```

in PSM where v1 is not updatable and v1 contains a column named c1 or c2 causes the unexpected error -12202 (i.e. Table V1 does not exist.)

This will no longer happen.

Type Resolution Problem for COALESCE in PSM (V9.2.2C)

Using `COALESCE(data,'')` combined with concatenation in PSM could cause unpredictable behavior.

This has now been corrected.

Date Arithmetic Problem in PSM (V9.2.2C)

The construction

```
(dateExpression1 - dateExpression2) intervalQualifier
```

could cause unpredictable behavior when used in PSM.

This has now been corrected.

On-line Backup (V9.2.1F)

A problem with collations and on-line backups have been corrected. The problem could sometimes prevent tables with collations to be accessed after an on-line backup.

Integrity Constraints and Triggers (V9.2.1C)

A table having both triggers and either FOREIGN KEY or UNIQUE constraints could previously allow updates that violated the integrity constraint.

This problem is now corrected.

Remaining Triggers (V9.2.1C)

Due to a error in the implementation for `DROP TABLE`, triggers referencing the dropped table could be left in data dictionary or the `DROP TABLE` statement would cause an unwarranted error. This happened if there were more than one trigger referencing the table being dropped.

This problem is now corrected.

Incorrect SQLSTATE in PSM (V9.2.1C)

After a `SELECT INTO` in PSM, the SQLSTATE was erroneously set to `02000` (instead of the correct `00000`) when a row was found.

This is now corrected.

CASE Statement in Scope of Continue Handler (V9.2.1C)

In some circumstances the presence of a continue handler and a `CASE` statement in a compound statement in PSM could cause unpredictable behavior.

This is now corrected.

Dropping VARYING Length Column Using ALTER TABLE (V9.2.1B)

An `ALTER TABLE` dropping a column could previously produce an inconsistent or corrupt table. The problem would only occur if the dropped column have a `VARYING` length data type and is followed by a `NATIONAL CHARACTER VARYING` column.

This problem is now corrected.

Scalar String Functions and Parameter Markers (V9.2.1)

When parameter markers were used for one or several arguments to a scalar string function, there were problems mixing arguments of the character data types with arguments of the national character data types. This is now corrected.

SUM Combined With GROUP (V9.2.1)

The result of a complex GROUP BY statement could previously return an incorrect result. The error did only occur for statements including a join with a table having a multi-column index.

The result of a `SUM` combined with a `GROUP` yields different results depending on which columns that are included in the select list. This is now corrected.

UPPER/LOWER and the German Character ß (V9.2.1)

The German character ß was not translated by the UPPER and LOWER functions in version 9.1 of Mimer SQL. In version 9.2, ß is treated as 'ss' and correctly translated.

Dropping Idents (V8.2.4)

When an ident was dropped while one or more objects created by that ident were in use then some of those objects might remain in the data dictionary. This made it impossible to recreate those objects. This has now been corrected.

Domains, Check Clauses and UPDATE STATISTICS (V8.2.4)

A domain with a check clause contained in a schema definition could not be used after an UPDATE STATISTICS statement.

The following sequence of statements:

```
create schema s create domain d int check(value <>0);
create table t(c1 s.d);
update statistics;
insert into t values(1);
```

would cause the following error:

```
Mimer/DB error -12504
Statement not allowed within transaction
```

The only way to avoid this was to drop the domain. This has been corrected.

Function SUBSTRING in PSMs (V8.2.4)

Use of the function substring in a PSM statement with an expression used as length yields an incorrect result. In some cases, some extra space characters were added to the result. This has now been corrected.

LEAVE LABEL and Atomic Compound Statement (V8.2.4)

Use of a leave label statement in an atomic compound statement caused an implicit rollback statement. This has now been corrected.

Qualified Function Reference in Set Statement (V8.2.4)

Use of a qualified function reference with a parameter marker in a set statement caused an internal inconsistency error. This has now been corrected.

PSM Routines and Triggers (V8.2.4)

In some cases, when creating PSM routines or triggers, not all dependencies were stored. This meant that the DROP with CASCADE option did not remove all depending objects. This has now been corrected.

Updating Using Next Value of Sequence (V8.2.4)

Using the `next_value of sequence_name` function in an update statement could cause data corruption or error messages such as 'The rowid column can not be updated'. This has now been corrected

ALTER TABLE and Constraints (V8.2.4)

In earlier versions of Mimer SQL 8.2, there was a problem when adding a foreign key or unique constraint using `ALTER TABLE`. This problem is now corrected.

Two rows with the same unique key value will now be allowed if at least one of the column values in the unique key is null. Rows with `NULL` column values in foreign keys will no longer have to reference a valid key.

In earlier versions of Mimer SQL 8.2, there was also a problem dropping a column that participated in a primary key constraint. The statement failed with an invalid record length error. This has now been corrected.

Columns and the WITH GRANT OPTION (V8.2.1)

It is now permissible to use the `WITH GRANT OPTION` with the statement `GRANT UPDATE` with specific columns

Longer Character-string Literals (V8.2.1)

The maximum length of a character-string literal is now increased to 15 000 characters.

Using DISTINCT in Views (V8.2.1)

Previously a view defined using a `SELECT DISTINCT` statement could return incorrect result sets. This has now been corrected.

ALTER TABLE and Dropping Columns (V8.2.1)

When dropping a column from a table, only views that actually use the dropped column will be dropped as a cascade effect. Previously, all views that used any column in the altered table would be dropped. This also applies to check constraints.

It is now possible to drop a column that is part of the primary key. The primary key constraint will be dropped as well.

Value of USER in Stored Procedure (V8.2.1)

During the execution of a stored procedure, the value of `CURRENT_USER` incorrectly was the same as `SESSION_USER`. It is now corrected so it is the name of the ident that created the procedure.

Support for GET DIAGNOSTICS (V8.2.1)

The following fields in the get diagnostics statements are now supported:

```
TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME, ROUTINE_CATALOG,
ROUTINE_SCHEMA,ROUTINE_NAME, CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
CONSTRAINT_NAME,CATALOG_NAME, SCHEMA_NAME, TABLE_NAME, COLUMN_NAME, NATIVE_ERROR
```

View with Subquery and Check Option (V8.2.1)

A view containing a subquery may now be specified with a `WITH CHECK OPTION` clause.

ABS Function for Interval (V8.2.1)

The function `ABS` now accepts an interval expression as a parameter.

Non-deterministic Check Constraint (V8.2.1)

A check constraint may not contain any reference to a possible non-deterministic expression, such as datetime value function and functions that are declared as not deterministic. This was previously not checked.

Grant on Added Columns (V8.2.1)

If a user has been granted a privilege on a table, he will get this privilege on any column that is added to the table.

Deallocating Statements in a Transaction (V8.2.1)

The server would previously hold all deallocated statements until the user's transaction was committed. This could require large amounts of SQLPOOL memory in large transactions, if many SQL statements were prepared and deallocated over and over again. The server no longer keeps more than a few deallocated statements.

Logical Expressions in PSM (V8.2.1)

Logical expressions such as `if not p or not q` then were not evaluated properly in a procedural context. This is corrected.

Loss of Significance PSM (V8.2.1)

In some cases with complex arithmetical expression using float data types, the error loss of significance would occur. This has now been corrected.

Scrollable Cursor and Union (V8.2.1)

It is now possible to use a scrollable cursor for a statement that contains a union operator.

Like Patterns in PSM (V8.2.1)

Procedures that used more than one `like` predicate (in PSM control statements) could crash the server. This has now been corrected.

Unique Constraints (V8.2.1)

If two users tried to insert the same value simultaneously in a column with a unique constraint this would not cause any constraint violation. This has now been corrected.

Time Problems (V7.3.x)

Certain time values (less than 10:00:00 that had been inserted using current_time) were not found when using an equal comparison. This problem has now been corrected.

Mimer ESQL Embedded SQL

Scrollable Cursor (V9.2.5)

If a scrollable cursor wasn't used directly after OPEN, so instead of a FETCH, another operation was performed, the exception "Invalid descriptor encountered on server" sometimes was raised. This is now corrected.

Setting Binary Descriptors With Character Data (V9.2.2E)

When setting a binary type descriptor with a null-terminated character data type using `SET DESCRIPTOR`, the string length was erroneously calculated from the maximum field length rather than looking for the terminating null character. In most cases, the user would experience a -14312 "Input character string too long" error. This problem was related to the client component and is corrected in version 9.2.2E.

Too Many Host Variables (V9.2.2D)

Using a huge number of host variables in one single SQL statement did not work correctly in previous versions. This problem is now corrected.

Mimer ODBC – Open Database Connectivity

Default Conversion for the SQL BIGINT Data Type Unavailable (V9.2.4)

The SQL data type `BIGINT` could not be fetched using the default C data type `SQL_C_DEFAULT`. Mimer ODBC clients prior to 9.2.4 encountered the error -109 "Invalid C data type" on such attempts. Fetching `BIGINT` columns using the `SQL_C_DEFAULT` data type will now correctly place the data in a 64 bit integer. In C this would correspond to a `long long int` or `unsigned long long int`.

Unknown Request Code in Network Package when Closing Cursors Against a Mimer 8.2.6 Server. (V9.2.3)

Applications opening cursors on a Mimer v8.2.6 server could experience the Mimer error -18551 error when closing the cursor (a call to `SQLFreeStmt` or `SQLCloseCursor`). The error text is "Unknown request code in network package". This error could also be seen when the ODBC driver decided to close the cursor for some other reason than the application calling these functions. The consequence of this problem is that cursors remain open on the 8.2.6 server until the client disconnects. This problem is corrected in v9.2.3 clients.

Conversion From Wide Character Parameter (V9.2.2G)

There has been a problem with conversions from `SQL_C_WCHAR` parameters to numeric data types. The problem can cause an “Invalid numeric value” error if the character string has several leading blanks. Note that the problem could occur when using Mimer SQL with external tools such as Microsoft Access.

This problem is now corrected.

ODBC Stand-alone Installation (V9.2.2E)

There has been a problem with Mimer ODBC stand-alone installations in previous versions of 9.2.

The installed Mimer ODBC driver could erroneously require the server to have a beta test licence. Note that the ordinary Mimer Installations are not based on ODBC stand-alone installation (see *Mimer Packaging Guide* for further information about ODBC stand-alone installation).

This problem is now corrected.

Disconnect Failure (V9.2.2E)

There has been a problem where `SQLDisconnect` fails with an “invalid transaction state” error. The problem occurred when `SQLDisconnect` is preceded by a failing data definition statement and the connection attribute `SQL_ATTR_AUTOCOMMIT` is set to `SQL_AUTOCOMMIT_OFF`.

This problem is now corrected.

Statement Increasing in Size Problem (V9.2.1C)

SQL statements supplied to `SQLPrepare` or `SQLExecDirect` when the statement attribute `SQL_ATTR_NOSCAN` is set to `SQL_NOSCAN_OFF` and whose length is less than 400 characters, but as a result of the escape clause processing grows to a size larger than 400 characters may return a -16183 (bad parameter) error. This problem was introduced with the version 9.2.1 server and is corrected in version 9.2.1C and later.

Support of SQL_ATTR_FETCH_BOOKMARK_PTR (V9.2.1B)

In earlier versions the attribute `SQL_ATTR_FETCH_BOOKMARK_PTR` was not supported by Mimer ODBC. Mimer ODBC does not currently support bookmarks at all, and any attempt to enable them returns an error. Nevertheless, from v9.2.1b onwards, the bookmark attribute `SQL_ATTR_FETCH_BOOKMARK_PTR` is itself supported, but the driver will not do anything useful with this.

Also, when an application tries to enable bookmarks using `SQLSetStmtAttr`(..., `SQL_USE_BOOKMARKS`,...), earlier versions of the Mimer ODBC driver returned `HY009` (invalid option value), rather than the more correct `HYC00` (option value not supported). This is now corrected.

Incorrect Capability Information for Forward Only Cursors (V9.2.1B)

Earlier versions of Mimer ODBC incorrectly reported that forward only cursor could do absolute and relative cursor positioning. This was incorrect. Forward only cursors can only move forward one row at a time in a result set. This problem applies only to applications coded and compiled according to the ODBC 3.0 specification.

Transaction That Lives On in Autocommit Mode When a BLOB, CLOB or NCLOB Operation Fails (V9.2.1)

When in auto commit mode a `BLOB`, `CLOB` or `NCLOB` operation fails in some way (such as a truncation error) after the transaction was started, the transaction lived on causing problems such as "xxx cannot be dropped because it is in use". This could also lead to that server resources in time become exhausted. The problem is now corrected when connecting to v9.1.3 servers or later.

Transaction That Lives On in Autocommit Mode When a Forward Only Cursor is Limited by the SQL_ATTR_MAX_ROWS Attribute (V9.2.1)

When a forward only cursor was limited in size by the `SQL_ATTR_MAX_ROWS` attribute, the ODBC driver now closes the cursor immediately when the last row was read. Previously, the cursor was kept open until the statement was explicitly dropped. This led to unnecessary server resource use. The problem applied to 8.2 servers or later, and the problem is now corrected against all server versions.

Auto-Commit of Read-Only Cursors (V9.2.1)

There has been a problem with server transaction resources associated with auto-committed read-only cursors. Previously, these resources could remain allocated after close cursor, until the cursor was opened again or the statement handle was freed. It could have a negative effect on server performance when such transaction resources remained allocated for a very long time. This problem is now corrected.

MFC and Mimer ODBC Include Files (V9.1.3)

When using Microsoft Foundation Classes (MFC) and the Mimer ODBC include file `MINODBC.H` compilation errors occurred. This was due to the fact that MFC defined certain symbols which interfered with the proper compilation of `MINODBC.H`. This has now been corrected.

MINODBC.H and MFC (V9.1.3)

MINODBC.H included conditional checks for definitions of `__SQLEXT` and `__SQLUCODE`. MFC based applications include header files that explicitly `#define` `__SQL` and `__SQLEXT` and this excluded large parts of the contents of MINODBC.H. These checks have now been removed.

SQLColAttributes driver specific attributes (V8.2.4D)

The Mimer ODBC driver returned an `-119 (unknown internal data type, SQLSTATE S1000)` error when the application called `SQLColAttributes` in ODBC 2.5 mode with a driver specific attribute.

It now returns `-151 (driver not capable, SQLSTATE S1C00)` since applications may rely on that error code to determine if the attribute was supported or not.

SQL_DESC_NAME and SQL_DESC_LABEL Regarded as Equal (V8.2.4)

Formerly, the descriptor attributes `SQL_DESC_LABEL` and `SQL_DESC_NAME` were always available to determine the column name and column labels of a statement.

For example, a `SELECT A AS B FROM C` involves a column A and a label B. From an SQL standards point of view, a label may be used to replace a name of a column for use within the query.

To comply with the SQL standard on this matter, the ODBC driver will now always regard `SQL_DESC_NAME` and `SQL_DESC_LABEL` as equal.

If a label is not specified, both will contain the name of the column. If a label is specified, both will contain the name of the label.

Problem with V8.2.1– 8.2.3 Clients (V8.2.4)

If the application called `SQLGetData` several times to get data into any other C-type than `SQL_C_CHAR` or `SQL_C_BINARY`, then the second call onward erroneously returned `SQL_NO_DATA_FOUND`.

Now, it returns `SQL_SUCCESS` with the correct data.

This was primarily a problem with version 8.2.1-8.2.3 clients. Earlier clients did not support getting columns in arbitrary order, only strictly ascending.

Arrayed Procedure Calls in Auto-commit Mode (V8.2.4)

A problem with arrayed procedure calls in auto-commit mode has been corrected.

Previously, when an arrayed procedure call terminated in an error, and auto-commit mode was on, the transaction was left in an active state. The transaction would normally terminate with the next statement, or `SQLEndTran` call.

Time and Timestamp Column Truncation (V8.2.1)

When converting TIME and TIMESTAMP columns with decimals to character (`SQL_C_CHAR`) and the output is truncated, the terminating null byte is now handled properly.

Furthermore, the length returned is the actual length and not the truncated length.

Auto-commit Behavior (V8.2.1)

When Mimer ODBC is in auto-commit mode, each statement is executed as a separate transaction. When using a select cursor, however, the transaction is not committed until the cursor is closed. It is thus possible for a single client to have several active transactions in the server.

This allows Mimer SQL to behave exactly as expected by an ODBC application. Please note that a server of version 8.2 or later must be used! The exact behavior with older server versions is described in the Mimer SQL 8.1 release notes.

REMARKS Column in Result Sets (V8.2.1)

The REMARKS column, found in result sets returned by `SQLTables` and `SQLColumns`, now returns comments created with COMMENT ON.

Truncation of Date/Time Values (V8.2.1)

`SQL_ERROR` is now returned if significant parts of a date/time value are truncated when converting to a character data type.

Fetching Rowsets (V8.2.1)

`SQLExtendedFetch` now returns `SQL_ERROR` only when all rows in the returned rowset failed. If one or more rows are returned successfully, this routine returns `SQL_SUCCESS_WITH_INFO`. When all rows are returned successfully, these routines return `SQL_SUCCESS`.

This information applies to `SQLFetch` and the new ODBC 3.0 function `SQLFetchScroll`.

Note: `SQLExtendedFetch` is deprecated in ODBC 3.0. The Driver Manager will map `SQLExtendedFetch` calls to `SQLFetchScroll`.

Interval Data Types (V8.2.1)

`SQLColumns`, `SQLGetTypeInfo` and `SQLProcedureColumns` now return correct values for all interval data types.

SQLForeignKeys (V8.2.1)

`SQLForeignKeys` will now correctly return information about all foreign keys.

Previously, `SQLForeignKeys` would only return information about tables created by the current_user unless a PKTableOwner or FKTableOwner was explicitly specified.

CALL with Qualified Procedure Names (V8.2.1)

The ODBC escape clause for `CALL` now supports qualified procedure names.

SQLColAttribute and Intervals (V8.2.1)

`SQLColAttributes` will now set `SQL_DESC_PRECISION` and `SQL_DESC_DATETIME_INTERVAL_PRECISION` to default-values when `SQL_DESC_TYPE` is set to `SQL_INTERVAL` and `SQL_DESC_INTERVAL_CODE` is given a valid value.

Commit Affects Executed But Not Fetched Cursors (V8.2.1)

Up until version 8.1 commit and rollbacks closed those cursors that no rows have been fetched from. This required an application to execute those statements again.

This is no longer the case, commit and rollbacks affect only those cursors that the application has fetched at least one row from.

SQLMoreResults Causes a Premature Autocommit (V8.2.1)

An `SQLMoreResults` which returned `SQL_NO_DATA` and was immediately followed by `SQLFreeStmt(…, SQL_CLOSE)` no longer causes a premature autocommit.

TINYINT not Supported by CONVERT (V8.2.1)

The `SQL_TINYINT` data type is now supported by the scalar function CONVERT.

Mimer BSQL

Describe View Problem (V9.2.5)

When describing a view in BSQL with a definition length > 400 characters the definition would not be displayed.

This has now been corrected.

Read Input (V9.2.5)

Starting BSQL with a query such as

```
bsql - q "read 'file'"
```

where file contains a read input command would cause BSQL to crash. This have now been corrected.

Describe Result in Wrong Order (V9.2.2E)

When describing a routine whose definition exceeds 800 character, the source definition would sometimes be mixed up. This has now been corrected.

Using <tab> in Load Statement (V9.2.2E)

The use of white-space characters (`<tab>`, `<cr>`, `<lf>`) in a `LOAD`/`UNLOAD` statement caused unwarranted syntax errors. This has now been corrected.

Failed Procedure Calls With Input Followed by Output Parameters (V9.2.1)

When BSQL was used to call a procedure having input parameters before the output parameters in the parameter list, BSQL returned errors such as -10310 “invalid character value for cast” or -10203 “negative overflow occurred during data type conversion”. Only 9.1 BSQL versions are affected by this problem. Earlier versions worked well.

String Literal Usage (V9.2.1)

When reading input containing character literals spanning multiple lines without a terminating apostrophe, it will not be a valid literal. If you wish to have a literal that spans multiple lines, you should use the string concatenation operator, e.g.

```
INSERT INTO music_title VALUES(4711,
  'The long and winding ' ||
  'character literal that spans ' ||
  'multiple lines')
```

Describe Functions – New Client/Old Server (V9.1.2)

Previously, in some describe functions, there was a problem using a version 9.1 BSQL client against a version 8.2 database server. This is now corrected.

Describe Table (V8.2.1)

The describe table command now returns information about tables that use the described table as a foreign key.

Log Files (V8.2.1)

Using several log file commands during a session did not work correctly in so far as nothing was logged except in the first file. This has now been corrected.

Transactions and Connections (V8.2.1)

Using several connections in BSQL and switching between these with active transactions caused BSQL to lose the ability to determine whether a transaction should be autocommitted or not. This has now been corrected.

Mimer Utilities

MIMLOAD Error Message Problem (V9.2.5)

The Windows release of Mimer SQL v9.2.4C had an error in MIMLOAD, which meant error messages produced from MIMLOAD were not printed correctly. This has been corrected.

DBC Giving Errors for SQLDB After Crash (V9.2.4)

A control of the SQLDB databank after a server crash may signal errors. SQLDB, which is considered as a work databank always indicates "properly closed", which lead to the DBC program earlier was doing a bitmap control.

Comments Causing Misplaced Breakpoints (V9.2.2E)

When trying to use the PSM debugger on a routine containing comments the icons indicating possible breakpoints were not set correctly. This problem also affected the STEP functionality and display of variable values.This has now been corrected.

DBC Program Not Fulfilling Databank Control (V9.2.2E)

If the databank has more than one rootpage (happens when there are many tables) and the last table entry stored in a rootpage holds an empty table, the `DBC` program misses to verify tables in the rootpage following.

This is now corrected.

MIMLOAD and Views Using Sequences (V9.2.2E)

`MIMLOAD` did not include any sequence usage (`NEXT_VALUE` or `CURRENT_VALUE`) in the view definititions. This problem is now corrected.

MIMLOAD and LOB Data (V9.1.3)

`MIMLOAD` can now load and unload `LOB`s correctly.

Mimer UTIL

Importing Files (V8.2.3)

In versions earlier than 8.2.3, it was not possible to import files that had been exported by using version 8.1. This problem has now been corrected.

Upgrading

Default NULL (V9.2.2)

At upgrade of databases from version 8.1 or earlier, default values explicitly specified as NULL were not handled correctly. Caused error message "Undefined value found during data type conversion" at describe table. This is now corrected.

Invalid Record Length for Index Tables with Character Varying Columns (V9.1.3)

An error concerning recreation of 'index tables' holding columns with data type `VARCHAR` has been detected in the upgrade program. If a `VARCHAR` column is defined as other than the first column participating in the "index table" (INDEX, UNIQUE or FOREIGN KEY) or in the primary key clause, the record length stored in dictionary is not calculated correctly. The error message "Table xxx has invalid record length" is encountered when the base table is to be opened. The error in the upgrade program is now corrected.

If you have an old upgrade program and receive this error message a workaround afterwards is to use `DROP INDEX` and `CREATE INDEX` for an ordinary index. If it is a constraint of type unique or foreign key use `ALTER TABLE DROP CONSTRAINT` and `ALTER TABLE ADD CONSTRAINT`.

CREATE TABLE and Check Clauses (V8.2.4)

Some problems detected when upgrading databases from earlier versions have been corrected in version 8.2.4. Previously, if there were several check clauses using the same column name in a CREATE TABLE statement, the upgrade program stopped.

Previously, if an old CREATE TABLE statement had both check clauses and foreign key references, the upgrade program succeeded. However, after upgrading, when working with such a table, the following error was reported:

Name XXX in check clause not recognized as a column name of current table definition. This is now corrected.

Database Inconsistencies (V8.2.4)

If the old database had inconsistencies (parts of objects not dropped) due to earlier bugs when dropping objects the upgrade program stopped. This has now been corrected and the upgrade program now proceeds and ignores such parts.

Missing Column Privileges (V8.2.4)

Old grants of privileges (update or references) to specified columns could be missing after upgrading. This is now corrected.

Unrecognized Keywords from V 7 (V8.2.3)

When upgrading from Mimer SQL version 7, the upgrade would fail if one of the keywords listed below was used as a name in some create statements. This problem has now been corrected.

The `CREATE` statements that were affected were: `CREATE TABLE` with a check clause, `CREATE DOMAIN` with a check clause, and `CREATE VIEW`.

```
ALLOCATE ALTER BEGIN CALL CLOSE COMMIT CONNECT CREATE CURSOR DEALLOCATE DECLARE
DESCRIBE DISCONNECT DO DROP ELSEIF EXECUTE FETCH GET GRANT IF INNER INOUT INTERVAL
JOIN LEAVE LEFT LOOP MODULE ON OPEN OUT OUTER PREPARE PROCEDURE REPEAT RESIGNAL
RETURN REVOKE RIGHT ROLLBACK SIGNAL SQLEXCEPTION SQLSTATE SQLWARNING START TO UNTIL
USING WHILE
```

VMS Specific Features

Using MIMINFO with DCL Command Qualifiers (V9.2.3C)

In version 9.2.3B, the `MIMINFO` command could not be used with DCL-style command qualifiers. Only the Unix-style command switches was supported. Version 9.2.3C supports both styles.

Error Messages in MIMER.LOG From Disconnected Clients (V9.2.1)

An error message could sometimes be produced on the mimer.log file when a Windows client disconnected its Mimer session. Since this event was harmless, these messages are no longer produced in the log file.

Hang When Waiting for I/O Completion (V9.1.3)

If prefetch of pages was interleaved with exclusive operations on a page, two threads could both be waiting for the completion of the I/O in which case only one succeeded and the other thread caused a hang in the database server. This problem has now been corrected.

Stopping a Database Server with Local Users Connected (V8.2.4)

If a database server was stopped while local users were connected to the server, the database server could not be restarted.

In order to restart the server, all processes that had mapped the `MIMCCS_xxxx` global section had to be found and terminated to avoid the possibility that those processes could interfere with the global section used by the newly started server. This problem has now been corrected.

In V8.2.4, the server can always be restarted, even if old users were not logged out properly. New global sections will be used for the new server process.

Note that the old global sections will still remain in the system until the old client processes terminate. This situation can lead to increased global page usage.

Stopping a Database Server with Users Connected via TCP/IP (V8.2.4)

If a database server was stopped while users were connected with the TCP/IP protocol (or if a performance monitor was connected to the server), the server could receive an access violation leading to a server crash. This problem has been fixed.

MIMTCP Keeping a Connection Open (V8.2.4)

In some circumstances, the `MIMTCP` process could keep a connection open to a database server. Because of this, the database server could not be restarted. In order to restart the server, the `MIMTCP` process has to be stopped first. This problem has now been corrected.

In V8.2.4, the `MIMTCP` process will always close its connection to the database server within two minutes when the server is stopped.

UNIX Specific Features

miminfo -u and Crashing Server (V9.2.4)

There has been a problem where a `miminfo -u` command under rare circumstances could make a database server crash. The problem would only occur on LINUX or Mac OS X. This is now corrected.

Undeleted Files Under /var/tmp (V9.2.1)

When a database server was owned and executed from another user that root, temporary files could be left undeleted on `/var/tmp`, which could lead to connection attempt failures when filenames were reused. This is now corrected.

Error Messages in mimer.log From Disconnected Clients (V9.2.1)

An error message could sometimes be produced on the mimer.log file when a Windows client disconnected its Mimer session. Since this event was harmless, these messages are no longer produced in the log file.

Connection Handshake Using /var/tmp (V9.2.1)

Tru64: In the operating system Tru64 version 5, the process ID number was extended from 5 digits to 6 digits. Since the process id was part of connection handshake filename (for platforms using Unix Domain Sockets derived from BSD 4.3) that was limited to 14 characters, where `/var/tmp/<pid>` was used, the connection attempt failed. This is now corrected.

Logging Out Local Connections by mimcontrol -l (V9.2.1)

A list of currently connected users can be obtained by the command:

```
$ miminfo -u <database>
```

It is possible to use the `mimcontrol` command to log out individual connections by specifying the channel number of that connection:

```
$ mimcontrol -l <channel> <database>
```

Previous versions of Mimer SQL on Unix could not log out local connections by using the `mimcontrol` command. This has now been corrected.

If the connection is executing in the server while it is logged out by `mimcontrol`, it will receive a cancel notification. This will cause the request to complete earlier and to automatically log out.

Named Pipes in /var/tmp (8.2.4F)

Linux:	Earlier Linux clients created an entry in `/var/tmp` when a local connection was created to a database server. The name of the entry corresponded with the `pid` the client process. In some cases these entries were not properly deleted which could cause problems. In the current version, the clients do not create any new entries in directories when connecting to a database server.

mimlicense Usage in miminstall (V8.2.4F)

In `miminstall`, the use of `mimlicense` has changed slightly. Now the name of the key delivered with the distribution is `mimerkey.mcfg`. Previously, the extension was `.cfg`.

Also, now the result of the `mimlicense` operation is checked and an error is displayed if something fatal occurs. Earlier this was silently ignored.

mimunlink and man-pages (V8.2.4F)

Previously, man-pages were not unlinked by the `mimunlink` tool. This is now corrected.

TMPDIR Verification in Shell Scripts (V8.2.4F)

It has now been verified that a possible `TMPDIR` environment variable setting is a valid directory path.

Previously, if this was not true, various error message could be issued.

Pathlist with Repeated Paths (V8.2.4)

Linux:	At some Linux installations, there was a problem with environment variable path lists, e.g. `LD_LIBRARY_PATH`, becoming too long and paths were repeated in the list. Previously, this could occur in shell scripts as they were setting a desired path and then padding with the original value of the variable. The procedure could then be performed recursively. This behavior has been changed by using a new program, called `mimaddpath`, which verifies if the added path already exists.

Relocated Documentation Index (V8.2.4)

Earlier the PDF index for the documentation set was distributed in a specific `manind` directory. This directory has been removed.

The directory, now called `index`, and the PDF file `index.pdx` are located on the same directory level as the PDF manuals. This is convenient when using the documentation index.

The documentation index is now included in the RPM distribution of Mimer SQL.

The Compatability Library (V8.2.4)

A symbol table has been added to the compatibility library `compat.a`.

The compatibility library has also been updated to include the sequential I/O package from version 7, mainly to provide for a possibility to relink the `MIMER/PG` tool.

The mimadmin Tool (V8.2.4)

The `mimadmin` menu tool has been updated to handle database names in a case-insensitive way. Upper case is used.

The dbinstall Tool (V8.2.4)

The `dbinstall` tool has four new optional arguments when used in the silent mode.

It is now possible to specify the database server operator, and locations for the `transdb`, `logdb` and `sqldb` databank files by using a new set of arguments, i.e. arguments 4-7 for the `-s` switch. For example:

```
dbinstall [ -s database password home_directory [ transdb_directory
logdb_directory sqldb_directory operator ] ]
```

The mimstatln Program (V8.2.4)

The `mimstatln` program, used to follow links, is now updated so that it always returns an absolute path.

The mimdbfiles Tool (V8.2.4)

The `mimdbfiles` tool has been updated so that raw device files used for databank storage are reported.

The mimautoset Tool (V8.2.4)

The `mimautoset` tool has ben updated so that created links to the `init.d` directory are removed when using the `-u` switch.

Updated Error Messages (V8.2.4)

In previous versions, some error messages from the server returned `unknown` as the specification of the function observing the error. These messages have been updated.

Problem with Long Pathnames in mimhosts (V8.2.3)

Previously, when adding a `pathname` using the `mimhosts` command, existing `pathnames` could become corrupted if the new `pathname` exceeded 100 characters. This has now been corrected.

Unexpected Line Wrap (V8.1.3B)

Linux: During installation and administration of the Mimer SQL system, unexpected line wraps could be encountered when scripts in the distribution were asking the user for information in order to continue. This could happen when using other shells than Korn Shell. This is now corrected.

Wrong Error Message (V8.1.3)

In the following example, when trying to start a database server without having the `MIMER_HOME` environment variable set, the wrong error message was displayed:

```
# mimcontrol -s m81pelle
1999-10-12 10:46:39.77   <Error>
Error when deleting memory pool in database server
#
```

The correct error message is:

```
The environment variable MIMER_HOME must point to the MIMER distribution
```

Automatic inetd Daemon Re-start (V8.1.3)

Linux: During installation, there is an option to automatically let the `inetd` daemon be re-started to enable `/etc/inetd.conf` updates. This operation was not always performed successfully during Mimer SQL installations on Linux. The process ID for the daemon was not obtained correctly.

Single-user Mode Databank Pathlist (V8.1.3)

Earlier when having a `pathlist` defined for a local database in `/etc/sqlhosts` only databanks in the database server home directory could be located when executing in single user mode. This is now corrected.

Windows Specific Features

XA Library Registration (V9.2.5)

When service pack 2 of Windows was introduced it was necessary to register all libraries that can perform distributed XA transactions. This is now done automatically for the Mimer ODBC dynamic link library.

PSM Debugger Shortcut (V9.2.5)

The PSM debugger shortcut shows up properly even if Java is not installed.

Checking SYSDB Using Databank Check Utility (V9.2.2E)

The previous 9.2 versions of the Databank Check Utility (dbcw.exe) returned an ‘File locked by other user’ error when trying to check SYSDB.

This problem is now corrected.

Using the NT Performance Monitor on Older Servers (V8.2.4)

Previously, the Mimer SQL NT Performance Monitor did not work when connecting to a database server of an older version than the database server installed on the client where the Performance Monitor was running.

This problem is now corrected, with some implications. Counter values not known by the database server will always yield a value of 0. Also, the SQL Pool deallocations/sec will always be 0 when working against 8.1.2 servers even though the counter exists.

Default License Key during Reboot (V8.2.4)

The default license key was not installed when the following conditions existed:

1. The Mimer SQL client was installed for the first time.
2. The ODBC software needed a reboot due to locked files.
3. The installation was made from a packed exe-file containing the Mimer SQL distribution downloaded from the Internet.

This has now been corrected.

Event Log Registry Entries during Uninstall (V8.2.4)

NT: When uninstalling Mimer SQL completely from a computer, the registry keys for event log handling were not removed. This is done properly now.

Uninstalling Default Data Source (V8.2.1)

When uninstalling Mimer SQL, ODBC data sources are saved. When Mimer SQL is subsequently reinstalled, the data sources are recreated. This now works for the default data source as well.

The handling of default data sources in the Mimer Administrator is now working correctly in all cases.

Parallel MIMCONTROL Operations (V8.2.1)

Previously, if several `MIMCONTROL` commands were executed concurrently the server would only respond to one of them and the other(s) would time out. This has now been corrected.

Mimer SQL Mobile Specific Features

TCP Server Process on Pocket PC (V9.2.5)

The TCP server only handled one connection properly before going into a non-responsive state. Nothing was shown in the window of the TCP Server process. Clients experienced "Connection refused" errors when this occurred. This has now been corrected.

Delimited Identifiers in the Mimer Explorer (V9.2.1)

The Mimer Explorer can now handle statement names that require delimited identifiers correctly (i.e. mixed case and/or special characters).

Statement Types in Mimer Explorer (V9.2.1)

The Mimer Explorer can now display statements that has been created as scroll only (CREATE SCROLL STATEMENT).

Chapter 4 Deprecated Features and Functions

This chapter describes Mimer SQL features and functionality deprecated in previous versions.

Mimer SQL Structured Query Language

Unqualified Column Name in ORDER BY When Duplicated (V9.2.4)

If a column name exists in several tables in a query, it must always be qualified in an ORDER BY specification. Prior to v9.2.4 the following statement did work because only SELECT list items were allowed in the ORDER BY clause:

```
SELECT t1.*, t2.c2 FROM t1, t1 AS t2 ORDER BY c1;
```

But since the introduction of ORDER BY expressions in v9.2.4, the `c1` specification in the above ORDER BY clause is ambiguous, since any column in the tables can be specified, even if its not in the SELECT list.

To avoid this problem, qualify the columns in the ORDER BY clause, e.g.

```
SELECT t1.*, t2.c2 FROM t1, t1 AS t2 ORDER BY t1.c1;
```

MIMER Views (V8.2.1)

The MIMER system views have been made completely backwards compatible in version 8.2. This means that they only return objects with a length of 18 characters or less. The new limit in Mimer SQL is now 128 characters. Because of this, the MIMER views do not show objects that are longer than 18 characters. Furthermore, a table that has any column with a name longer than 18 characters will not be shown.

One effect of this is that old Mimer SQL clients only show objects that conform to the old limits.

Applications retrieving dictionary information should migrate to the `INFORMATION_SCHEMA` views. These views return all information, including any new information such as constraint names, as well as Mimer SQL specific information, such as comments and databanks.

SET TRANSACTION CHANGES (V8.1.1)

The statements `SET TRANSACTION CHANGES INVISIBLE` and `SET TRANSACTION CHANGES VISIBLE` are no longer needed. However, the statements are supported in version 8.2 for compatibility reasons.

Mimer ESQL Embedded SQL

INCLUDE SQLCA (V7.x.x)

In version 7.1 of Mimer SQL, it was necessary to use INCLUDE SQLCA in an embedded SQL program to include the declaration of the SQL communication area. INCLUDE SQLCA is no longer required.

Applications should now use the SQLSTATE variable and the GET DIAGNOSTICS statement to get all the information previously obtained from SQLCA.

See the *Mimer SQL Programmer's Manual* for a description of SQLSTATE and GET DIAGNOSTICS. For compatibility reasons, the use of INCLUDE SQLCA is still supported.

SQLCODE (V7.x.x)

The use of SQLCODE to retrieve status information was replaced by the use of SQLSTATE and GET DIAGNOSTICS in version 7.2 of Mimer SQL.

For compatibility reasons, return codes can still be retrieved in SQLCODE.

However, SQLCODE can be either a field in the SQLCA, or a 4-byte integer variable in the application. Mimer SQL will assume the existence of an SQLCODE variable in the application if: no INCLUDE SQLCA statement is found; and neither SQLSTATE nor SQLCODE has been declared between BEGIN DECLARE SECTION and END DECLARE SECTION.

The values of SQLCODE are the same as the values for the internal Mimer SQL return codes described in the Mimer SQL Programmer's Manual, Appendix C.

SQLDA (V7.x.x)

The SQL descriptor area SQLDA, which was used in version 7.1 of Mimer SQL, has been replaced by a standardized SQL descriptor area.

The SQLDA area was allocated and maintained by constructions in the host language. The new SQL descriptor area is allocated and maintained by standardized embedded SQL statements.

The old SQL descriptor area SQLDA is still supported in Mimer SQL for compatibility reasons.

BSQL

Load/Unload Deprecated (V9.1.1)

The BSQL functionality for `Load/Unload` is now deprecated. Use `MIMLOAD` instead.

Mimer Utilities

Mimer UTIL (V9.2.1)

UTIL is deprecated and has been removed from the Mimer SQL distribution.

In Mimer UTIL there was an option to generate special system databanks for the old Mimer modules FM, PG, QL, RG, and SH. Please, contact our support department if this option is required.