



Mimer SQL

Features and Functionality

History

Mimer SQL, Features and Functionality History
© Copyright Upright Database Technology AB.

The contents of this manual may be printed in limited quantities for use at a Mimer SQL installation site. No parts of the manual may be reproduced for sale to a third party.
Information in this document is subject to change without notice. All registered names, product names and trademarks of other companies mentioned in this documentation are used for identification purposes only and are acknowledged as the property of the respective company. Companies, names and data used in examples herein are fictitious unless otherwise noted.

Produced and published by Upright Database Technology 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	1
The Mimer SQL Database Server.....	1
PSM Debugger (V8.2.4).....	1
Mimer SQL Structured Query Language.....	2
Triggers (V8.2.1)	2
Instead-of Triggers (V8.2.1).....	2
PSM Functions (V8.2.1).....	2
Schema Support (V8.2.1)	2
Sequences (V8.2.1).....	3
Relaxed Rules for Views (V8.2.1).....	3
Updateable Primary Keys (V8.2.1).....	3
Named Constraints (V8.2.1).....	3
Recursive Procedures (V8.2.1).....	3
Record Data Type (V8.2.1).....	4
ATOMIC (V8.2.1)	4
UNDO (V8.2.1).....	4
128 Character Names (V8.2.1)	4
System View Support (V8.2.1).....	4
Like Expression (V8.2.1).....	5
Soundex Function (V8.2.1)	5
New Reserved Words (V8.2.1)	5
SESSION_USER (V8.2.1).....	5
UNIQUE Columns May Be NULL (V8.2.1).....	5
OVERLAPS (V8.2.1)	6
GRANT/REVOKE on Individual Column (V8.2.1).....	6
Revoke GRANT OPTION (V8.2.1)	6
COMMENT ON New Objects (V8.2.1).....	6
Optional AS in Select List (V8.2.1)	6
Explicit Defaults (V8.2.1).....	6
LOCALTIME, LOCALTIMESTAMP (V8.2.1)	6
SET Statement (V8.2.1).....	7
Binary Data Type (V8.2.1)	7
Alter Table (V8.2.1).....	7
On Delete Rule (V8.2.1).....	7
Get Diagnostics in PSM (V8.2.1)	7
Compiler Directives (V8.2.1).....	8
New Scalar Functions (V8.2.1)	8
Online Backup (V8.2.1).....	8

Mimer ESQL Embedded SQL	9
Support for New SQL Statements (V8.2.1).....	9
SQL Statement Classification (V8.2.1).....	9
Full SQL92 Support (V8.2.1)	9
Character Pointer Support (V8.2.1)	9
Mimer ODBC – Open Database Connectivity	9
ODBC Version 3 Support (V8.2.1).....	9
New ODBC Data Types (V8.2.1).....	9
New ODBC Data Types (V8.2.1).....	10
New ODBC Routines (V8.2.1).....	10
ODBC Escape Clause Enhancements (V8.2.1)	10
Row Wise Parameter Binding (V8.2.1).....	11
New Statement Options (V8.2.1)	11
Mimer BSQL	11
Support for New SQL Syntax (V8.2.1)	11
Delimited Data in LOAD/UNLOAD (V8.2.1).....	11
Mimer Utilities	11
Mimer UTIL	11
New Functions.....	11
UNIX Specific Features	12
Large Files Support (V8.2.4).....	12
Automatic Startup (V8.2.2).....	12
Man Pages (V8.2.2).....	12
RPM Support (V8.2.2)	12
JDBC Driver (V8.2.2).....	12
New Utilities – Version 8.2 (V8.2.2)	12
I/O Threads (V8.1.3).....	13
POSIX Threads (V8.1.1)	13
New Utilities – Version 8.1 (V8.1.1).....	13
Managing /etc/sqlhosts Using mimhosts (V8.1.1).....	15
Managing Client Connects Using mimitcp (V8.1.1).....	15
Symbolic Links to Mimer (V8.1.1)	15
Windows Specific Features	15
Getting Started after Reboot (V8.2.4).....	15
Automatic Server Restart (V8.2.1).....	16
New File Extensions (V8.2.1)	16
New Routines for ODBC Driver Configuration (V8.2.1)	16
Chapter 2 Changed Features and Functions	17
The Mimer SQL Database Server	17
Performance Enhancements (V8.2.4)	17
Faster Access with Long Read-only Transactions (V8.2.4)	17
Index Lookup Only (V8.2.1).....	17
New Reorganization Algorithm (V8.2.1)	18
Bitmap Lookup (V8.2.1).....	18
Databank Check in Background (V8.2.1)	18
Block Pre-fetch (V8.2.1)	18
Caching of Compiled Queries (V8.2.1).....	18

Optimized Sort/Load Operations (V8.2.1).....	19
Enhanced Work Table Management (V8.2.1).....	19
Optimized Aggregate Functions Expressions (V8.2.1)	19
Group by Optimization (V8.2.1).....	19
New Licensing System (V8.2.1).....	19
Mimer SQL Structured Query Language.....	19
Cascade/Restrict (V8.2.1).....	19
Privilege Changes (V8.2.1).....	20
No Duplicate Privileges (V8.2.1).....	20
Update Statistics Cleans Indexes (V8.2.1)	20
Representation of DOUBLE PRECISION and FLOAT (V8.2.1)	21
CREATE IDENT Changes (V8.2.1).....	21
CREATE DATABANK Changes (V8.2.1).....	21
Hex-constants (V8.2.1)	21
DDL-statements in Transactions (V8.2.1)	21
Mimer ODBC – Open Database Connectivity	22
Performance Enhancements (V8.2.1).....	22
Less Server Communication with ODBC (V8.2.1)	22
Automatic Read-only Cursors with Auto-commit (V8.2.1).....	22
SELECT FOR UPDATE (V8.2.1).....	22
SQLSTATE Changes (V8.2.1)	23
Changes to ODBC Routines (V8.2.1)	23
Truncate Without Warning by SQLFetch (V8.2.1).....	23
Mimer SQL Data Type FLOAT (V8.2.1).....	23
Extended SQLGetData Functionality (V8.2.1)	23
Mimer BSQL	24
The CONNECT Statement (V8.2.4)	24
List and Describe (V8.2.1)	24
Show Settings (V8.2.1)	24
Mimer Utilities	24
Mimer UTIL	24
Load/Unload Menu (V8.2.1).....	24
Using UTIL in a Script (V8.2.1)	24
Using INFORMATION_SCHEMA (V8.2.1).....	25
VMS Specific Features	25
Dump Files and MIMER.LOG (V8.2.4)	25
MIMINFO Command Syntax for Selecting a Database (V8.2.4)	25
MIMTCP Process (V8.2.2).....	25
Installation is Pre-linked (V8.1.1).....	25
No Single-user Mode Libraries (V8.1.1).....	26
New SINGLEDEFS.DAT Parameter (V8.1.1)	26
Changed Directory Structure (V8.1.1).....	26
Multi-vendor TCP/IP Support (V8.1.1)	26
Changed MIMSETUP8 Parameters (V8.1.1)	27
DECNET Object Name is the Database Name (V8.1.1)	27
No Files on SYS\$SHARE (V8.1.1).....	27
MULTIDEFS.DAT (V8.1.1)	27
UNIX Specific Features.....	28
The DumpScript Parameter in the multidefs File (V8.2.4F)	28

Automatic Server Dump Facility (V8.2.4)	28
Updated Error Tracing (V8.2.4)	28
Root Not Required (V8.2.2)	28
Example makefile Version 8.2 (V8.2.2)	29
Licensing System (V8.2.2)	29
SDBGEN (V8.2.2)	29
Default Number of Users (V8.2.2)	29
tmp Not Used (V8.2.2)	29
Client Channel Limit (V8.1.3D)	29
Asynchronous I/O Limit (V8.1.3D)	30
MemLock Default in multidefs (V8.1.3D)	30
Menu System, mimadmin (V8.1.3)	30
mimunlink (V8.1.3)	30
MIMER_HOME Not Required (V8.1.3)	30
Single-user Shared Library Lookup	31
Database Server Startup	31
Example makefile Version 8.1 (V8.1.3)	31
Database Server Alert Messages (V8.1.2)	31
Administration Environment (V8.1.1)	32
No Single-user Mode Programs (V8.1.1)	32
Changed Directory Structure (V8.1.1)	32
DB Server Configuration File, multidefs (V8.1.1)	33
Database Registry File, /etc/sqlhosts (V8.1.1)	33
TCP Ports for Database Servers (V8.1.1)	33
Database Server Aliases (V8.1.1)	34
Database Server Dump Directory (V8.1.1)	34
ODBC Support (V8.1.1)	34
Windows Specific Features	34
Mimer Administrator (V8.2.1)	34
Mimer Info Enhancements (V8.2.1)	34
DB-check Enhancements (V8.2.1)	35
Chapter 3 Corrected Features and Functions	37
The Mimer SQL Database Server	37
Next Values of Sequences (V8.2.4)	37
Online Backup Problem (V8.2.4E)	37
Background Thread Loop (V8.2.4E)	37
Multiple Updates of Same Record with Secondary Index Access (V8.2.4E)	37
Invalid Timestamp Messages During online Backup (V8.2.4E)	38
Creating Temporary Tables Using MIMER/PG (V8.2.4E)	38
Many Request Threads and High Load (V8.2.4)	38
Growing T-cache (V8.2.4)	38
Error Handling and Low Memory (V8.2.4)	38
DDL Statements and Older Clients (V8.2.4)	38
Mimer ESQL Clients (V8.2.4)	39
Commit Set Problems (V8.2.4)	39
Accessing Views in Read Only Transactions (V8.2.3)	39
Mimer SQL Structured Query Language	39
Dropping Idents (V8.2.4)	39
Domains, Check Clauses and UPDATE STATISTICS (V8.2.4)	39
Function SUBSTRING in PSMs (V8.2.4)	39

LEAVE LABEL and Atomic Compound Statement (V8.2.4).....	40
Qualified Function Reference in Set Statement (V8.2.4).....	40
PSM Routines and Triggers (V8.2.4)	40
Updating Using Next Value of Sequence (V8.2.4)	40
ALTER TABLE and Constraints (V8.2.4).....	40
Columns and the WITH GRANT OPTION (V8.2.1).....	40
Longer Character-string Literals (V8.2.1)	40
Using DISTINCT in Views (V8.2.1).....	40
ALTER TABLE and Dropping Columns (V8.2.1)	41
Value of USER in Stored Procedure (V8.2.1)	41
Support for GET DIAGNOSTICS (V8.2.1).....	41
View with Subquery and Check Option (V8.2.1).....	41
ABS Function for Interval (V8.2.1).....	41
Non-deterministic Check Constraint (V8.2.1).....	41
Grant on Added Columns (V8.2.1)	41
Deallocating Statements in a Transaction (V8.2.1).....	41
Logical Expressions in PSM (V8.2.1).....	41
Loss of Significance PSM (V8.2.1).....	42
Scrollable Cursor and Union (V8.2.1)	42
Like Patterns in PSM (V8.2.1).....	42
Unique Constraints (V8.2.1)	42
Time Problems (V7.3.x)	42
Mimer ODBC – Open Database Connectivity	42
SQLColAttributes driver specific attributes (V8.2.4D).....	42
SQL_DESC_NAME and SQL_DESC_LABEL Regarded as Equal (V8.2.4)	42
Problem with V8.2.1– 8.2.3 Clients (V8.2.4)	43
Arrayed Procedure Calls in Auto-commit Mode (V8.2.4).....	43
Time and Timestamp Column Truncation (V8.2.1)	43
Auto-commit Behavior (V8.2.1).....	43
REMARKS Column in Result Sets (V8.2.1)	43
Truncation of Date/Time Values (V8.2.1)	43
Fetching Rowsets (V8.2.1)	43
Interval Data Types (V8.2.1).....	44
SQLForeignKeys (V8.2.1).....	44
CALL with Qualified Procedure Names (V8.2.1)	44
SQLColAttribute and Intervals (V8.2.1).....	44
Commit Affects Executed But Not Fetched Cursors (V8.2.1)	44
SQLMoreResults Causes a Premature Autocommit (V8.2.1).....	44
TINYINT not Supported by CONVERT (V8.2.1).....	44
Mimer BSQL	44
Describe Table (V8.2.1).....	44
Log Files (V8.2.1).....	45
Transactions and Connections (V8.2.1).....	45
Mimer Utilities	45
Mimer UTIL.....	45
Importing Files (V8.2.3).....	45
Upgrading	45
CREATE TABLE and Check Clauses (V8.2.4)	45
Database Inconsistencies (V8.2.4)	45
Missing Column Privileges (V8.2.4)	45
Unrecognized Keywords from V 7 (V8.2.3).....	45

VMS Specific Features	46
Stopping a Database Server with Local Users Connected (V8.2.4).....	46
Stopping a Database Server with Users Connected via TCP/IP (V8.2.4).....	46
MIMTCP Keeping a Connection Open (V8.2.4).....	46
UNIX Specific Features	47
Named Pipes in /var/tmp (8.2.4F).....	47
mimlicense Usage in miminstall (V8.2.4F).....	47
mimunlink and man-pages (V8.2.4F).....	47
TMPDIR Verification in Shell Scripts (V8.2.4F).....	47
Pathlist with Repeated Paths (V8.2.4).....	47
Relocated Documentation Index (V8.2.4).....	47
The Compatability Library (V8.2.4).....	48
The mimadmin Tool (V8.2.4).....	48
The dbinstall Tool (V8.2.4).....	48
The mimstatln Program (V8.2.4).....	48
The mimdbfiles Tool (V8.2.4).....	48
The mimautoset Tool (V8.2.4).....	48
Updated Error Messages (V8.2.4).....	48
Problem with Long Pathnames in mimhosts (V8.2.3).....	48
Unexpected Line Wrap (V8.1.3B).....	49
Wrong Error Message (V8.1.3).....	49
Automatic inetd Daemon Re-start (V8.1.3).....	49
Single-user Mode Databank Pathlist (V8.1.3).....	49
Windows Specific Features	49
Using the NT Performance Monitor on Older Servers (V8.2.4).....	49
Default License Key during Reboot (V8.2.4).....	49
Event Log Registry Entries during Uninstall (V8.2.4).....	50
Uninstalling Default Data Source (V8.2.1).....	50
Parallel MIMCONTROL Operations (V8.2.1).....	50
Chapter 4 Deprecated Features and Functions	51
Mimer SQL Structured Query Language	51
MIMER Views (V8.2.1).....	51
SET TRANSACTION CHANGES (V8.1.1).....	51
Mimer ESQL Embedded SQL	52
INCLUDE SQLCA (V7.x.x).....	52
SQLCODE (V7.x.x).....	52
SQLDA (V7.x.x).....	52

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

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

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_SCHEMA` – 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 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

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

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 ' ';
```

Mimer Utilities

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.

UNIX Specific Features

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 `mimauto` and `mimserver` 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
<code>dbfiles</code>	Lists the databank files for a server, as stored in the data dictionary.

Utility	Function
mimautoreset	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 dbserver 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
asctoexp	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.	

Program	Short explanation	Used by
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 <code>/etc/sqlhosts</code> file. See <i>Managing /etc/sqlhosts Using mimhosts (V8.1.1)</i> on page 15 for more information.	dbinstall, mimadmin
miminfo	Program to monitor database servers. Replaces the old mimserv program.	mimadmin
mimlink	Command used to link Mimer SQL libraries and executables to <code>/usr/lib</code> and <code>/usr/bin</code> , respectively. See <i>Symbolic Links to Mimer (V8.1.1)</i> on page 15 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 <i>Managing Client Connects Using mimtcp (V8.1.1)</i> on page 15 for more information.	
mimuninstall	Command to uninstall Mimer SQL version 8.	
mimunlink	Command used to remove symbolic links from <code>/usr/bin</code> and <code>/usr/lib</code> , previously created by the <code>mimlink</code> command.	mimuninstall

Program	Short explanation	Used by
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

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 `mcfq`) are registered when the software is installed.

`mcfq` 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.

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

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

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` ident. These programs have now been replaced by system privileges:

Old <code>PROGRAM</code> 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 <code>EXECUTE</code> on MIMER_BR to ident revoke <code>EXECUTE</code> on MIMER_BR from ident	grant <code>BACKUP</code> to ident revoke <code>BACKUP</code> from ident
grant <code>EXECUTE</code> on MIMER_SC to ident revoke <code>EXECUTE</code> on MIMER_SC from ident	grant <code>STATISTICS</code> to ident revoke <code>STATISTICS</code> from ident
grant <code>EXECUTE</code> on MIMER_SW to ident revoke <code>EXECUTE</code> on MIMER_SW from ident	grant <code>SHADOW</code> to ident revoke <code>SHADOW</code> 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

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

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

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 28, 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: /dl/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/libmimdb.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 (`net_srvr`) 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.

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

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

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 ODBC – Open Database Connectivity

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 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

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

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

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

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 been 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

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.

Chapter 4

Deprecated Features and Functions

This chapter describes Mimer SQL features and functionality deprecated in previous versions.

Mimer SQL Structured Query Language

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.