MaxDB/liveCache Development Support
February 2010
Expert Session
New Features in SAP MaxDB Version 7.7

MaxDB/liveCache Development Support
Heike Gursch
Oksana Alekseious
February 16, 2010
**Agenda**

1. **New concepts in 7.7**
   - Database Studio, Parameter Names, KnlMsg
   - Multiple Log Partitions
   - I/O Interface
   - Multiple Snapshots, etc.

2. **Other Kernel Features**
   - Length Restriction for Records
   - Create Index, etc.

3. **DBM-Server**
   - Background Commands
   - SDBMAIL Integration into Event Dispatcher

4. **Other Changes**
   - ODBC, JDBC, etc.
This document describes major changes between MaxDB 7.7 and smaller versions. For details please refer to the (online) documentation. Some features have been implemented in version 7.6 as well.

Even if some of the features described below are not directly perceptible, they might have a strong impact, e.g. on performance or scalability, and thus are briefly described.

**Terminology remark**

Within this document LOBs (Large Objects, also known as BLOB or CLOB) are mentioned. The MaxDB data type LONG has been renamed to LOB. The deprecated SQL syntax using LONG still works. Access to the catalog views shows the types BLOB and CLOB.
Database Studio

The Database Studio is an Eclipse based Java tool substituting the DBMGUI and SQLStudio. It further has GUI components like loader plug-ins and supports database servers as of version 7.5. The Database Studio can run on all supported platforms. For now it’s released for Windows and Linux (both 32 and 64 bit).
As of version 7.7.03 the parameter names were consolidated. Therewith most parameters got a new name without containing underlines. The legibility of parameter names is improved by the use of upper and lower case characters. You can read and set the parameters by using the old names. The command `param_directgetall` only shows the new parameter names. The view `ACTIVECONFIGURATION` shows old and new parameter names.

<table>
<thead>
<tr>
<th>PARAMETERNAME</th>
<th>PERMANENT</th>
<th>CHANGEABLE</th>
<th>DEPRECATED</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CacheMemorySize</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>5000</td>
</tr>
<tr>
<td>CACHE_SIZE</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>5000</td>
</tr>
</tbody>
</table>
The file `KnlMsg` contains messages of the database kernel. It is recreated each time the database instance is started. The previous file is renamed to `KnlMsg.old`. The messages - apart from the header (start messages) - are overwritten cyclically.

Error messages are recorded in `KnlMsg` but also - due to the risk that they will be overwritten there - in the file `KnlMsgArchive (knldiag.err)`. This file is written continuously.

As of version 7.7 `KnlMsg` files replace the files `knldiag*`. A specialty of the new files is that they are stored in an XML-like representation to make it possible in further states of expansion that together with the error messages directly instructions are delivered. This implies that the files – if you look at them on operating system level – have to be prepared before they can be displayed properly (`protconv`). If you choose Database Studio, DBMGUI or transaction DB50 to display the `KnlMsg` the conversion to a readable format is done automatically.

```xml
<MSG _NO="4618" _PROCESS="948" _THREAD="0x16DC" _TIME="2007-06-10 18:38:11.638">
  <MSG _NO="1" _ID="20232" _COMP="RTE" _TEXT="Dump of all kernel parameters start">
    <MSG_ARGS
      _FILE="RTEConf_ParameterAccessKernelInterface.cpp",
      _LINE="1131",
      _TIME="2007-06-10 18:38:11.622",
      _DETAILS="744",
    />
  </MSG>
</MSG>
```
Database Studio offers to the user to either display the file KnlMsg in the familiar classical way or in the XML representation (see above). By double-clicking a line in the XML representation you can get more information about the error (see next slide).
The following windows are displayed delivering more information about the error and proposing possibilities to correct the error. As mentioned above the windows are still partially empty and some more content is required.
MaxDB builds clusters for tables with the cluster flag to improve read performance for scans. If blocks are written for cluster tables the pager tasks are looking for logically clustered blocks. Logically clustered blocks are those with successive cluster keys. The cluster key is defined by the primary key or another logical key which must not be unique on application side (f.e. time characteristic). Pager tasks write those blocks adhesively to the data area.

A cluster built by pager tasks is only written to a separate FBM (Free Block Manager) section if the number of blocks within the cluster is at least ClusterWriteThreshold % of DataIOClusterSize and a free section in the data volumes is available. During backup and restore the clustering is not lost. If the percentage falls below ClusterWriteThreshold and no more free section is available the cluster is splitted and written to different free blocks.

If the database is filled to a high amount there is increased risk of writing too small clusters because there are no more free FBM sections for bigger clusters. So the scan performance of the system will be restricted.

FBM sections are released if they are only filled with a few blocks and if the condition for parameter ClusterCompressionThreshold is fulfilled.

At the end of a savepoint it is checked by pager tasks if there are FBM sections with a low filling grade. Server tasks read the affected blocks to the data cache and mark it as modified. The blocks are written to other positions in the data area at the latest with the next savepoint. The FBM sections are now free for large table clusters.
As of version 7.7 MaxDB allows the use of multiple log partitions. With parallel writing to the log volumes the database prevents bottlenecks during the access to the log queue and additionally wait situations for writes into the log volumes.

Partitions and also volumes of the partitions may have different sizes.

Normally user tasks of a UKT are assigned to a special log writer and therewith to a partition. This implies that for some tasks the state „log full“ might occur even if there is still some free space for a user task of another UKT in the corresponding log partition. Perform a backup of the log area if the state „log full“ is shown.

With the use of the CLEAR LOG command the backup history is interrupted. Make sure that a complete data backup is performed. If desired the automatic log backup can be switched on again.

The concept has been implemented in version 7.7 but productive use is only recommended as of version 7.8.
Automatic log backup is not a new feature of 7.7.
The database kernel can create the log backup automatically.
You activate automatic log backup with the dbmcli command autolog_on.
The log backup is automatically created asynchronously upon completion of a segment.
Newly implemented in 7.7 is the use of a time interval; there has also been a down port to 7.6.
As of MaxDB version 7.6.02 also a time interval may be set to launch the automatic log backup along this interval.

Autosave log allows setting a maximum online time between two log backups. dbmcli sets the interval with medium_put or with autosave_on in seconds.

```
dbmcli => medium_put "LOGBACKUP" "/dbarchive/savelog" FILE AUTO 0 8 NO NO "" NONE 600
dbmcli => autosave_on LOGBACKUP
```
From version 7.5, you can freeze the data area of a database instance using a snapshot.

In versions 7.5 and 7.6 a snapshot is generated in the ADMIN state. As of 7.7 it is also possible to create it in ONLINE mode. Later you can reset the data to its state at the time of the snapshot and/or delete the snapshot.

With the CREATE_SNAPSHOT command, the database kernel copies the restart page from the second block of the first data volume to another position. The complete converter is also copied. The original restart record contains a reference to the restart record that corresponds to the snapshot.

With the command RESTORE_SNAPSHOT, the current converter is deleted. All blocks that are no longer needed are marked as free in the FBM (Free Block Manager). The log is formatted such that the state HISTLOST occurs. At the next restart, the instance works with the data as they were at the time of the CREATE_SNAPSHOT.

The statement DROP_SNAPSHOT deletes the restart record and the corresponding converter that is relevant for the snapshot. The FBM marks all blocks that are no longer needed as free.

Up to 7.6 MaxDB supports only a single snapshot, as of 7.7 several snapshots can be generated. Operating the instance with one or several snapshot(s) uses more of the capacity of the data area.
MaxDB version 7.7 is able to administer several snapshots at the same time. You can create and drop snapshots in online mode.

An instance (reader) can access to the snapshot of another instance (provider) via the I/O interface and import tables logically.

Access to the snapshot is done in read-only mode. Changed blocks of the imported tables are stored physically in the data volumes of the reader.

In that way using a master system a lot of system copies on the level of tables or schemas, respectively, can be created. The required space of the reader is basically determined by the changed blocks.

The readers subject to the usual concept for MaxDB backups whereas only those blocks are saved that are stored in the reader. For a restore the snapshot of the accordant provider must be accessible.

The import of a snapshot and the related tables also works within the same instance if a second schema is used.
With version 7.7 the I/O interface to the operating system has been reimplemented. Version 7.7 uses different parameters than version 7.6. The new I/O system in version 7.7 has the following essential advantages:

- No direct assignment of an I/O worker thread to a volume. This implies a better scalability of I/O.
- I/O worker threads can be started on request. This prevents the use of unnecessary resources.
- The synchronization of accesses to the I/O queues has been changed. The access is done collision free. This additionally improves the scalability of I/O.
- Prioritization of special I/O requests. Dedicated jobs within the database (e.g. CHECK DATA) can run with lower priority. Online operation is stressed less.
- Tasks can send I/O requests asynchronously to the I/O system. They don't have to wait until the I/O request has been fulfilled but can continue their work.
- Support of multiple database instances.
Legacy I/O Components in MaxDB <= 7.6

I/O Workers with Queues

Example:
- Two queues per volume
  (_IOPROCS_PER_DEV = 2)
- Two requests in queue before switch to next
  (_IOPROCS_SWITCH = 2)
- Three jobs started in three tasks in parallel
Transition to new I/O –
Make I/O Queues Lock-free

Volume Job Queues
Lock-free SIAO

I/O Workers

© SAP AG 2010. All rights reserved / Page 18
New I/O Components in MaxDB >= 7.7

Volume Job Queues
Lock-free SIAO

First job in queue, generate ticket

Second job in queue, no ticket

Execute I/O

Return workers

Ticket in queue, wake up worker

I/O done
The shared memory of the runtime still exists after emergency shutdowns. The MaxDB console can start a database kernel which is able to read all information from the shared memory although the productive database is not available anymore. This improves the analysis of database aborts.

```
myserver:e70adm> x_cons E70 sh act
SERVERDB: E70

ID UKT UNIX TASK APPL Current Timeout Region Wait
  tid type  pid state priority cnt try item
T215 7 21117 User 0* Running 0 1446 10 18(r)

*** Post Mortem Analysis for ServerDB E70 using kernel ***

```
Multi Version Concurrency Control (release postponed to version 7.8)

If the isolation level is set to 'committed read' MaxDB now can read data without share locks. MaxDB keeps track of older versions of data items until they are no longer involved in any open transaction.

More detailed information will be delivered soon.
MaxDB now supports records with a length of up to 32767 bytes. The maximum length for a single column still is 8100 bytes. If a record's size exceeds 8100 bytes, a part of it is stored in a chain of 8KB pages. The record itself contains the start page number of that page chain.

→ redirection implies performance impact for usage of large records
→ in system view FILES, record extension pages will be shown as part of TREELEAVESSIZE
SQL statements now may utilize up to 10000 host variables (instead of 2000).

Especially in automatically generated commands the value was exceeded.

2000 host variables were not enough in WebAS applications.

As of 6.40 MaxDB 7.6.06.02, 7.7.06.08, 7.7.07.04 and 7.8 now support 10000 input parameters.

The limit of the DBSL can be increased to 10000 with the SAP instance profile parameter dbs/ada/input_parameters when the relevant MaxDB database release is used.

The limit for the number of output parameters is still 1000.
CREATE INDEX runs concurrently to normal operation on the base table. It does not lock the base table exclusively for a longer period. Modifications on the base table are logged during index construction. They are applied to the index at the end of the procedure.
Index maintenance is space and time consuming. It does not make sense to define too much indexes that are not really used by the optimizer. To check whether and how often an index was used you can select in DOMAIN.INDEXES (column INDEX_USED).

MaxDB counts each usage of a secondary key index. The database can reset the index usage counter with the following command:

```
ALTER INDEX <indexname> on < schemaname >.< tablename > INIT USAGE
```

The INDEX_USED column in the system table DOMAIN.INDEXES is initialized with 0 then. This means that the mechanism that counts how frequently an index is used is restarted.
Query Rewrite

Complex statements will be rewritten to achieve a better and more efficient optimization and execution. Heuristic rules are applied, qualifications should be applied to the base table directly, even if placed somewhere else in the original statement.

<table>
<thead>
<tr>
<th>RULENAME</th>
<th>ACTIVE</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 AddLocalPredicates</td>
<td>YES</td>
<td>Add Local Predicates for Joins with OR-Predicates</td>
</tr>
<tr>
<td>2 ConvertExistentialSubquery</td>
<td>NO</td>
<td>Convert a correlated existential subquery to an IN clause</td>
</tr>
<tr>
<td>3 ConvertOrToIn</td>
<td>YES</td>
<td>Convert OR to IN</td>
</tr>
<tr>
<td>4 ConvertTeExistentialSubquery</td>
<td>YES</td>
<td>Convert INTER SECT or EXCEPT to an existential subquery</td>
</tr>
<tr>
<td>5 DistinctForSubqueries</td>
<td>YES</td>
<td>Set Distinct for existential and all subqueries</td>
</tr>
<tr>
<td>6 DistinctPullUp</td>
<td>YES</td>
<td>Remove distinct elimination in a select if all fromselects are distinct</td>
</tr>
<tr>
<td>7 DistinctPushDownFrom</td>
<td>YES</td>
<td>Distinct push down from</td>
</tr>
<tr>
<td>8 DistinctPushDownTo</td>
<td>YES</td>
<td>Distinct push down to</td>
</tr>
<tr>
<td>9 EliminateGroupByOrDistinct</td>
<td>YES</td>
<td>Remove unnecessary GROUP BY or DISTINCT</td>
</tr>
<tr>
<td>10 EliminateOrderBy</td>
<td>YES</td>
<td>Remove unnecessary ORDER BY</td>
</tr>
<tr>
<td>11 EliminateSubqueries</td>
<td>YES</td>
<td>EliminateSubqueries</td>
</tr>
<tr>
<td>12 MergeExistentialSubquery</td>
<td>YES</td>
<td>Merge a select with an existential subquery</td>
</tr>
<tr>
<td>13 MergeFromSelectOrView</td>
<td>YES</td>
<td>Merge a select with a fromselect or view</td>
</tr>
<tr>
<td>14 NormalizePredicates</td>
<td>YES</td>
<td>Normalize Predicates</td>
</tr>
<tr>
<td>15 OptimizeSubqueries</td>
<td>YES</td>
<td>OptimizeSubqueries</td>
</tr>
<tr>
<td>16 PushDownPredicates</td>
<td>YES</td>
<td>Push down predicates</td>
</tr>
<tr>
<td>17 PushDownProjection</td>
<td>YES</td>
<td>Push down projection</td>
</tr>
<tr>
<td>18 PushDownQuantifier</td>
<td>NO</td>
<td>Push down quantifier</td>
</tr>
<tr>
<td>19 RemoveConstFromGroupOrOrderBy</td>
<td>YES</td>
<td>Remove unnecessary constants from GROUP BY or ORDER BY</td>
</tr>
<tr>
<td>20 SimplifyPredicates</td>
<td>YES</td>
<td>Simplify Predicates</td>
</tr>
</tbody>
</table>

Query Rewrite is not a new functionality of 7.7 but has been improved in comparison to 7.6.
Unicode Column Compression

MaxDB internally stores Unicode data in the UTF-16/UCS-2 format. This requires twice as much storage space as saving the data in ASCII format, since the coding is 16-bit rather than 8.

Now, MaxDB supports a compression for non key columns which reduces the space requirements for Unicode data. The Unicode Column Compression can be used for the entire database or just for single tables.

Database Parameter: UseUnicodeColumnCompression

ALTER TABLE [<schemaname>].<tablename> [NOT] PACKED
You can use the LIMIT clause (limit_clause) to limit the maximum number of rows in the result table. You either enter only the desired maximum number of rows or the additional information from which row the list should begin (offset value). The offset value of the initial row is 0. If no offset value is specified, the rows are listed from the beginning of the result table.

If both a LIMIT clause and an ORDER clause are entered, then all result rows are sorted and the relevant number of rows is displayed. Normally, the result differs from what you would receive if no ORDER clause was entered.

You can use the TOP syntax element to specify that only the first $n$ lines of the result are to be generated. Whole numbers between 0 and 2147483647 are permitted.

If you use a TOP syntax element and an ORDER clause in a SELECT statement, the first $n$ lines of all lines sorted by the ORDER clause are generated. If you do not use an ORDER clause in the SELECT statement, any $n$ lines are generated.

If a QUERY expression (query_expression) consists of several QUERY specifications (query_spec), the TOP syntax element must only be contained in the first QUERY specification.
MaxDB now supports the combination of UPDATE and INSERT.

It’s an implementation of the following sequence

```
if record found
    then update record
else
    insert record
```

An application can avoid one communication step with the database when using UPSERT.
MaxDB now allows modifications on values with the data type LOB using literals.

```sql
CREATE TABLE mylob
(col1 CHAR (10),
 colc CLOB,
 colb BLOB,
 PRIMARY KEY (col1))

INSERT INTO mylob (col1, colc, colb)
VALUES ('First_Key', 'This can be a very long value', x'00000000000000000000000000000000')
```
DELETE WITH COMMIT

This command will execute the delete operation and subsequently commit it independent of the
transaction it runs in. It only works for tables without referential integrity constraints and without
trigger definitions. The insert will not be rolled back even if the contextual transaction will be. It
might be useful e.g. for writing logs to keep track of your application and extends the INSERT
WITH COMMIT introduced with version 7.6.

```sql
count = 0;
while (sqlcode != 0)
{
    INSERT INTO tab (... VALUES (...);
    count++;
    if (count mod 1000 == 0)
        DELETE FROM log_table WHERE key = 'myentry'
            WITH COMMIT
    }
if (sqlcode != 0)
    COMMIT WORK
```
Enhanced ORDER BY for UNION

MaxDB supports an ORDER BY clause for columns in the column list of the first select in a union query expression.

```
SELECT dummy FROM dual
UNION
SELECT 1 FROM dual
ORDER BY dummy
```
As alternative to the traditional user authentication, database users can now be created in a new fashion.

With this method a user can connect to the database either through his operating system user or through one of the security protocols Kerberos or Secude.

<table>
<thead>
<tr>
<th>CREATE USER e70adm IDENTIFIED EXTERNALLY AS 'e70adm'</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; myserver:e70adm 302&gt; sqlcli -d E70 -u e70adm select dummy from dual</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>1 row selected (436 usec)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ALTER USER sape70 IDENTIFIED EXTERNALLY AS 'e70adm'</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; mysesrver:e70adm 302&gt; sqlcli -d E70 -u sape70 select dummy from dual</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>1 row selected (436 usec)</td>
</tr>
</tbody>
</table>
You can define whether a database user can open database sessions.

If CONNECT mode is not specified, ENABLE CONNECT is implicitly assumed. ENABLE and DISABLE cannot both be specified at the same time.

**ENABLE CONNECT**: The database user (members of the database user group) can open database sessions.

**DISABLE CONNECT**: The database user (members of the database user group) cannot open database sessions. ALTER USER|USERGROUP statement: The database sessions of the specified database user (members of the specified database user group) are terminated. You can enable the individual members of a database user group to open database sessions again using the ALTER USER statement.
New Role DBA

All DBA users take over the privileges of the system role DBAROLE

```
sqli cli firstdb=> GRANT SELECT ON firstdb.zztele TO dbarole

sqli cli anotherdb=> SELECT * FROM firstdb.zztele

...```
New View SYSDDLHISTORY

The system view SYSDDLHISTORY shows all DDL commands and the timestamps of the command executions.

<table>
<thead>
<tr>
<th>SCHEMNAME</th>
<th>TABLENAME</th>
<th>EXECUTED AT</th>
<th>STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST</td>
<td>TEST0</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST0 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST1</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST1 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST2</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST2 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST3</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST3 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST4</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST4 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST5</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST5 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST6</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST6 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST7</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST7 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST8</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST8 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST9</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST9 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST10</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST10 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST11</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST11 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST12</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST12 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST13</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST13 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST14</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST14 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST15</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST15 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
<tr>
<td>TEST</td>
<td>TEST16</td>
<td>2007-10-02 10:40:52.352</td>
<td>CREATE TABLE TEST16 (COL1 CHAR(10), COL2 CHAR(10),...</td>
</tr>
</tbody>
</table>

Please note: SYSDDLHISTORY is a regular table and will grow over time and might take up a lot of space in the database.
The MaxDB Database Manager Server now supports background commands. Those commands are mainly used by the Database Studio.

Earlier long-running DBM Server commands like e.g. backup and restore blocked a DBM Server session and were bound to it irreversibly. Now we can start a long-running command in the background DBM Server which can be taken over to another DBM Server session.

The new DBM server background commands are:

- `background_server_execute <bg_server_name> [-no_reply] <command>`
- `background_server_exit <bg_server_name>`
- `background_server_get_reply <bg_server_name> [<-skip_bytes>]`
- `background_server_show_status <bg_server_name>`
- `background_server_start <bg_server_name>`
- `background_server_takeover <bg_server_name>`

Let’s switch to a demo …

Short description of the demo:
- first window: logon with dbmcli
- second window: logon with dbmcli
- second window: start background server BACKUP, util_connect (no_reply), start the data backup
- first window: dbm_shm_info, takeover of the background server, get reply, util_release (no_reply) and exit
- second window: show status of the background server, check with dbm_shm_info
Preparations done for the demo:
Create dbm_whiteList.cfg with entry sdbmail

dbm_configset SDBED_SMTPSENDERAD <email address of sender>

dbm_configset SDBED_SMTPSERVERS mail.sap.corp
Preparations done for the demo:

```
event_dispatcher ADD Name == Offline Command == "sdbmail \"<subject>Database EXPERDB went into OFFLINE mode\</subject> <body>Database EXPERDB went into OFFLINE mode Date:$EVTDATE$ Time:$EVTTIME$</body><recipient>name@sap.com</recipient>
```

Short description of the demo:
```
event_list
```

Switch the database to OFFLINE mode.

⇒ Automatic email is generated.
The ODBC Driver has been re-implemented and uses SQLDBC. The new driver runs with much better performance. Especially bulk operations have been accelerated significantly.

The new ODBC driver can work without the MS ODBC driver manager.
Improvements of the JDBC driver regarding performance:

- The MaxDB JDBC driver now is able to use variable length of input and output variables.
- Positioned fetches in result sets have been accelerated significantly.
- The driver provides methods for fast access to catalog data belonging to result sets.
DBM-Server supports “Portable MaxDB Installation”

With version 7.7 MaxDB supports U3 environments. A database can run without any footprint on the database server (usually desktops).

The DBM client and server are able to run without any registrations.
Pipe Based Communication for JDBC Powertoys

The MaxDB JDBC powertoys enable JDBC applications to use DBM commands for database administration and loader commands for export and import jobs.

The powertoys now support a pipe based communication for local connections. With this new communication type the JDBC driver remains a poor java interface which increases the stability of the interface. Application crashes cannot abort the Java VM.
You have learned about the most important changes in version 7.7

- Some more or less formal changes have been done: new parameter names, new format of KrNLMsg, switch to new administration tool Database Manager
- A new I/O concept has been introduced.
- Clustering of tables and indexes accelerates larger scan operations.
- Multiple log partitions are developed.
- Multiple snapshots are possible.
- Some other helpful kernel changes:
  - Lock free index generation
  - Unicode column restriction
  - Other user authentication possible
  - etc.

- New dbmserver functionality

- Some changes concerning ODBC and JDBC
Questions and Answers
March 10, 2010

Session 7: SAP MaxDB Software Update Basics