



# Q-MONITOR v3.4.1

for Q-CHECKER®

## INSTALLATION GUIDE



## Orientation Symbols Used in the Manual

For better orientation in the manual the following symbols are used:

### Warning triangle



The warning triangle refers to *critical circumstances*, which should be considered *imperatively* in order to avoid *serious* problems in your work.

### Hint symbol



The light bulb relates to *hints*, which provide you with practical examples to simplify your work.

### Note symbol



The hand symbol relates to *notes*, which you should pay attention to in order to assure that you can *work without problems*.

### Info symbol



The info symbol relates to background *information*.

### Step symbol



The work steps symbol refers to a *step-by-step instruction* sheet.

TRANSCAT in the Internet:

<http://www.transcat-plm.com>

Q-MONITOR in the Internet:

<http://www.q-checker.com/>

Q-MONITOR Hotline:

Phone: +49 721 970 43 100

E-mail: [q-checker@transcat-plm.com](mailto:q-checker@transcat-plm.com)

© TRANSCAT PLM GmbH & Co. KG, 2008

## Table of Contents

1.	Prerequisites .....	4
1.1	Prerequisites for Q-MONITOR Application.....	5
1.2	Prerequisites for Q-MONITOR Applet.....	5
1.3	Prerequisites for Q-CHECKERDB .....	5
2.	Files in the Delivery .....	6
3.	Installation Scenarios .....	7
3.1	Standard Installation .....	7
3.2	Applet Installation .....	8
4.	Installation Procedure .....	9
4.1	Installing Q-MONITOR Application.....	9
4.1.1	Installation on UNIX.....	10
4.1.2	Installation on WINDOWS.....	12
4.2	Database Installation .....	13
4.2.1	Database Structure .....	13
4.2.2	Create Database Tables .....	14
4.2.3	Initial Filling and Updating of Criteria Table.....	17
4.2.4	Test Connection with Q-MONITOR .....	19
4.3	Administration of Q-CHECKER with JDBC Storing .....	20
4.3.1	UNIX/CATIA V4.....	20
4.3.2	UNIX/CATIA V5 .....	22
4.3.3	WINDOWS/CATIA V5 .....	25
4.4	Administration of Q-CHECKER with FTP Storing .....	28
4.4.1	UNIX/CATIA V4.....	28
4.4.2	UNIX/CATIA V5 .....	32
4.4.3	WINDOWS/CATIA V5 .....	36
4.5	Q-MONITOR Applet.....	40
5.	Administrating Q-MONITOR .....	43
5.1	Database Structure .....	43
5.2	Initialization File * .ini .....	44
5.2.1	DB_INFO .....	46

## TABLE OF CONTENTS

---

5.2.2	SQL_INFO .....	47
5.2.3	OPT_INFO .....	49
5.2.4	GEN_INFO.....	52
5.2.5	HELP_INFO .....	54
5.3	Description of Q-CHECKER's XML-Parameters .....	55
5.3.1	DB_INFO Section .....	55
5.3.2	CHECK_SESSION Section.....	56
5.3.3	CHECK_CRITERION Section.....	57
6.	Unicode Enabling.....	59
7.	Trouble Shooting .....	60

\* \* \*

# 1. Prerequisites

Q-MONITOR is designed to evaluate Q-CHECKER check results that have been saved in a relational database. You must therefore have such database and a TCP IP connection to the database server.

- Supported Database Management Systems:   ⇒ DB2  
  ⇒ Oracle  
  ⇒ MS SQL Server  
  ⇒ mySQL  
  ⇒ PortgreSQL
- Other SQL databases on request.

## 1.1 Prerequisites for Q-MONITOR Application

- JAVA 1.4 or above
- JDBC driver for database access

## 1.2 Prerequisites for Q-MONITOR Applet

- JAVA browser plug-in 1.4 or above on client
- JDBC driver for database access on client
- Web server for applet distribution on server

## 1.3 Prerequisites for Q-CHECKERDB

- JAVA 1.4 or above
- JDBC driver for database access

In case of FTP storing:

- FTP server

## 2. Files in the Delivery

The Q-Monitor software package consists of the following files:

- `qmonitor.x.x.x.jar`  
This file contains JAVA classes of Q-MONITOR
- `qmonitor.bat`  
Startup batch file for WINDOWS
- `qmonitor.ksh`  
Q-MONITOR Startup script for UNIX
- `qMon32.ico`  
Desktop icon for WINDOWS
- `Q-Monitor_readme.txt`  
Latest program modifications of Q-MONITOR
- `qmonitor.ini`  
Initialization file for Q-MONITOR; contains the information for the database connection, report definitions, location of the help files
- `qserver.html / index.html`  
Example for Q-MONITOR usage as applet
- `certificate.crt`  
TRANSCAT-PLM-Browser certificate for Q-MONITOR usage as applet
- `doc/Q-Monitor_Installation_Guide.pdf`  
Q-MONITOR Installation and Administration Guide
- `doc/Q-Monitor_User_Guide.pdf`  
Q-MONITOR User Guide
- `Q-CheckerDB.x.x.x.jar`  
This file contains JAVA classes of Q-CHECKERDB
- `Q-CheckerDB_Server.ksh`  
Q-CHECKERDB Startup script for UNIX
- `Q-CheckerDB_Server.vbs`  
Q-CHECKERDB Startup script for WINDOWS
- `Q-CheckerDB_readme.txt`  
Latest program modifications of Q-CHECKERDB

## 3. Installation Scenarios

Q-MONITOR can be used and installed in different scenarios. In each sample scenario different applications are to be installed. In the following a description will be given of two different installation cases—standard installation and applet installation.

### 3.1 Standard Installation



The standard installation is a local installation of Q-MONITOR. The Q-CHECKER check results from the user computers are stored in the database via JDBC.

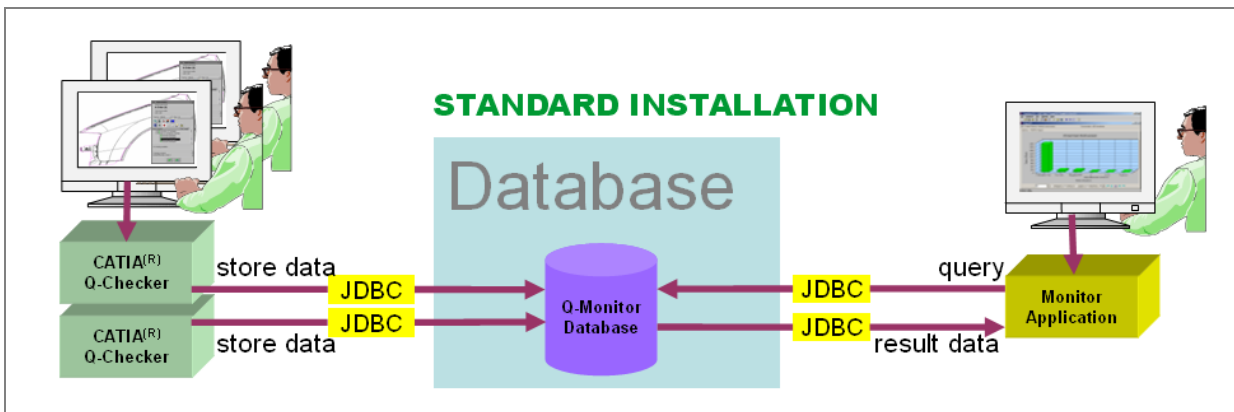


Figure 3-1: Dataflow in case of standard installation

The standard installation requires the following steps:

- (1) Install Q-MONITOR application—see section 4.1.
- (2) Create database tables—see section 4.2.
- (3) Administration of Q-CHECKER with JDBC storing—see section 4.3.

## 3.2 Applet Installation



In case of the applet installation Q-MONITOR is installed on a WEB server, but not locally on the user computers. The Q-CHECKER check results from the user computers are send via FTP to the database server and stored there in the database. The user can access Q-MONITOR, using a browser applet.

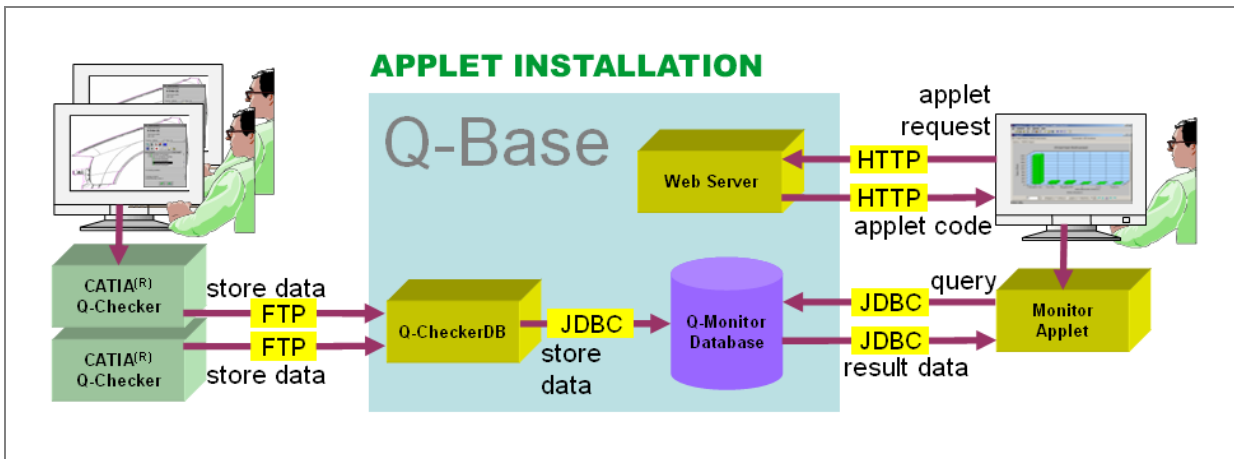


Fig. 3-2: Dataflow in case of applet installation

The applet installation requires the following steps:

- (1) Install Q-MONITOR Application—see section 4.1.
- (2) Create database tables—see section 4.2.
- (3) Administrate Q-CHECKER with FTP storing—see section 4.4.
- (4) Install Q-MONITOR applet—see section 4.5.



**NOTE:**

Database and WEB server have to run on the same machine because the applet is not allowed to connect to another machine then the WEB server.

## 4. Installation Procedure

For installation scenarios refer to chapter 3.

### 4.1 Installing Q-MONITOR Application

- **General: Adapt start-up file**

To run Q-MONITOR you must adapt the startup file corresponding to your operating system:

- `qmonitor.bat` (for WINDOWS)
- `qmonitor.ksh` (for UNIX)

These files contain the commands to start the JAVA Virtual Machine (JVM) and to set the JAVA classpath. The JAVA classpath must be set such that the JVM can find the following JAVA classes:

Classes	File name	Explication
RTE base	<code>rt.jar</code>	<p>These classes are for the most part contained in the <code>rt.jar</code> file in the JAVA installation directory under <code>.../lib</code>.</p> <p>Generally, it is necessary to pass to JVM the path to the JAVA Runtime Environment (RTE) base classes (<code>rt.jar</code>). Under WINDOWS, in the majority of cases, this is not necessary. On UNIX systems, it is necessary in every case.</p>
Q-MONITOR	<code>qmonitor.x.x.x.jar</code>	The Q-MONITOR software itself
JDBC driver	File name depends on database. These classes are also contained in a <code>*.jar</code> or in a <code>*.zip</code> file.	Software classes for JAVA database access. These files must be copied from the database installation directory.

### 4.1.1 Installation on UNIX

Load the Q-MONITOR software package on your system and extract the file into a desired directory. To run Q-MONITOR on UNIX, you must adapt the passages marked **gray** in the `qmonitor.ksh` file to the realities of your local Q-MONITOR installation.

```
#!/usr/bin/ksh
#-----
# Q-Monitor start script
#-----
QMONITOR_APPL="/qmonitor.3.x.x.jar"
QMONITOR_JDBC="/jdbc_driver.zip"
#-----
Sys=`uname -s`
if [ "$Sys" = "AIX" ]
then
  JAVAPATH=/usr/java14
  CLASSPATH="${JAVAPATH}/jre/lib/rt.jar:${QMONITOR_APPL}:${QMONITOR_JDBC}"
elif [ "$Sys" = "HP-UX" ]
then
  JAVAPATH=/opt/java1.4
  CLASSPATH="${JAVAPATH}/jre/lib/rt.jar:${QMONITOR_APPL}:${QMONITOR_JDBC}"
elif [ "$Sys" = "IRIX" ] || [ "$Sys" = "IRIX64" ]
then
  JAVAPATH=/opt/java1.4
  CLASSPATH="${JAVAPATH}/jre/lib/rt.jar:${QMONITOR_APPL}:${QMONITOR_JDBC}"
elif [ "$Sys" = "SunOS" ]
then
  JAVAPATH=/opt/java1.4
  CLASSPATH="${JAVAPATH}/jre/lib/rt.jar:${QMONITOR_APPL}:${QMONITOR_JDBC}"
fi
#-----
${JAVAPATH}/bin/java -classpath ${CLASSPATH} qmon.QMonitor ./qmonitor.ini
#-----
```

Specification of the variables:

Variable name	Explication
QMONITOR_APPL	Path and name of the Q-MONITOR classes
QMONITOR_JDBC	<p>Path and name of the JDBC driver. The JDBC driver is delivered with the database, e.g. for</p> <ul style="list-style-type: none"> <li>• DB2: db2java.zip</li> <li>• Oracle: classes12.zip</li> <li>• PostgreSQL: postgresql-xxx.jar</li> <li>• MySQL: MYSQL.jar</li> <li>• MSSQL Server: MSSQL.jar</li> </ul>
JAVAPATH	Installation path of JAVA (Location is operating-system dependant)
./qmonitor.ini	If the qmonitor.ini file has a non-standard location, such as in a user home directory, its complete path must be passed to the JVM.

## 4.1.2 Installation on WINDOWS

To run Q-MONITOR on WINDOWS, you must adapt the passages marked gray in the `qmonitor.bat` file to the realities of your local Q-MONITOR installation.

```
@echo off
REM -----
REM Q-Monitor start script
REM -----
set QMONITOR_APPL=".\\qmonitor.3.x.x.jar"
set QMONITOR_JDBC=".\\jdbcdriver.zip"
set JAVAPATH="C:\\Program Files\\Java\\j2re1.4.2\\bin\\java.exe"
set CLASSPATH="C:\\Program Files\\Java\\j2re1.4.2\\lib\\rt.jar;%QMONITOR_APPL%;%QMONITOR_JDBC%"
REM -----
%JAVAPATH% -classpath %CLASSPATH% qmon.QMonitor .\\qmonitor.ini
REM -----
```

Specification of the variables:

Variable name	Explication
QMONITOR_APPL	Path and name of the Q-MONITOR classes
QMONITOR_JDBC	Path and name of the JDBC driver. The JDBC driver is delivered with the database, e.g. for <ul style="list-style-type: none"> <li>• DB2: db2java.zip</li> <li>• Oracle: classes12.zip</li> <li>• PostgreSQL: postgresql-xxx.jar</li> <li>• MySQL: MYSQL.jar</li> <li>• MSSQL Server: MSSQL.jar</li> </ul>
JAVAPATH	Installation path of JAVA
CLASSPATH	Library path of JAVA
\\.\\qmonitor.ini	If the <code>qmonitor.ini</code> file has a non-standard location, such as in a user home directory, its complete path must be passed to the JVM.

## 4.2 Database Installation

To generate the database tables, an SQL script defining the specific tables must be written. This SQL script can be easily created with the Q-MONITOR user interface.

The database/instance has to be created before by the database administrator.

### 4.2.1 Database Structure

The database in which check results are saved must consist of three tables:

- CHECK\_SESSION
- CHECK\_CRITERION
- CRITERION\_INFO

The CHECK\_SESSION table contains columns for general information from the checks. This information is, for example, model names, model size, date of check, user name, etc. In this table there is only one entry for each check.

In the CHECK\_CRITERION table, information about the criteria checked is saved. This table has as many entries for each check, as there are criteria checked. The exact number depends on the current check profile. From this table you can get information as to whether a criterion was violated or not, how many times it was violated, and what priority it has.

The CRITERION\_INFO table contains the names of the criteria in different languages.

Figure 4-1 shows the three tables, their columns and how they are related.

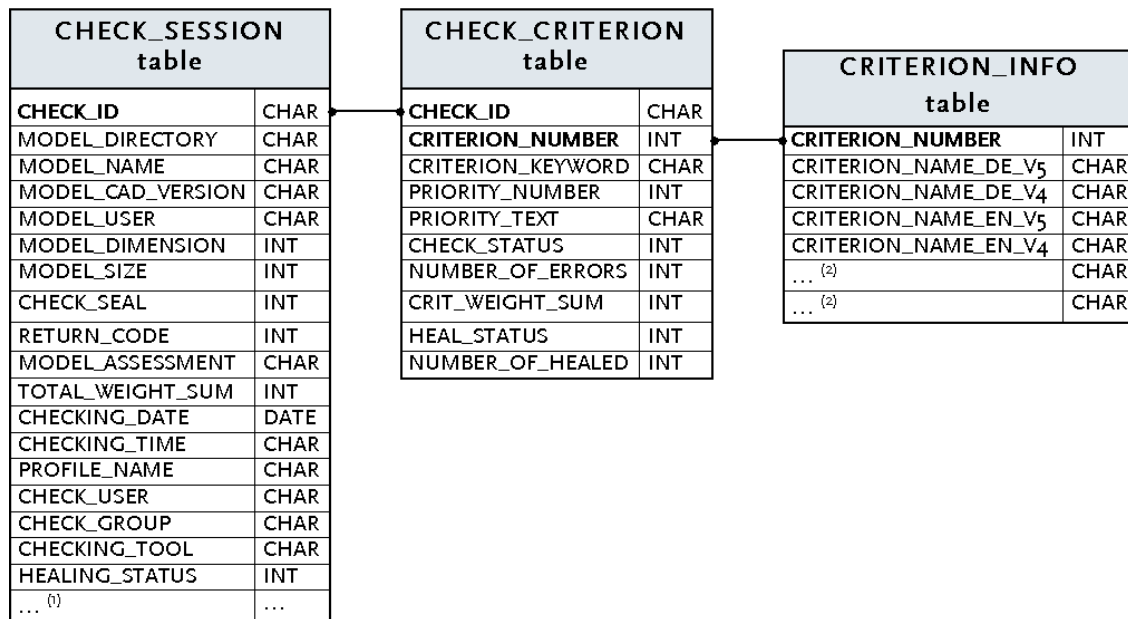


Figure 4-1: Database structure

Explication:

- (1) There can be additional, user-defined columns.  
 (2) Depending on the languages defined by the user, there can be additional columns for other languages.

Abbreviations used for data types: CHAR—character, INT—integer

## Restriction Options

The number of restriction options is the same as the number of fields in the database record (i.e. columns in the CHECK\_SESSION table). This means that the test results can be filtered using any of their attributes.

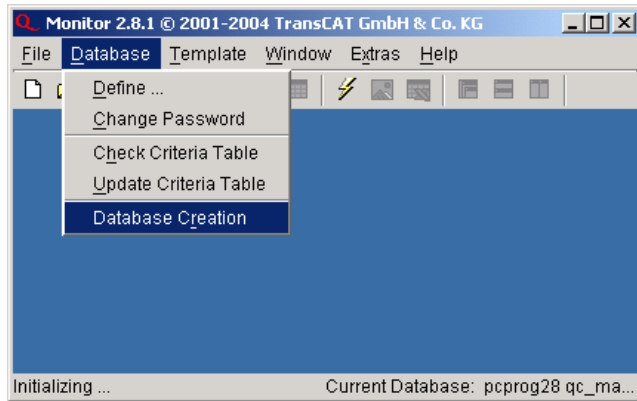
### 4.2.2 Create Database Tables

To create the SQL script for the creation of database tables do the following steps.

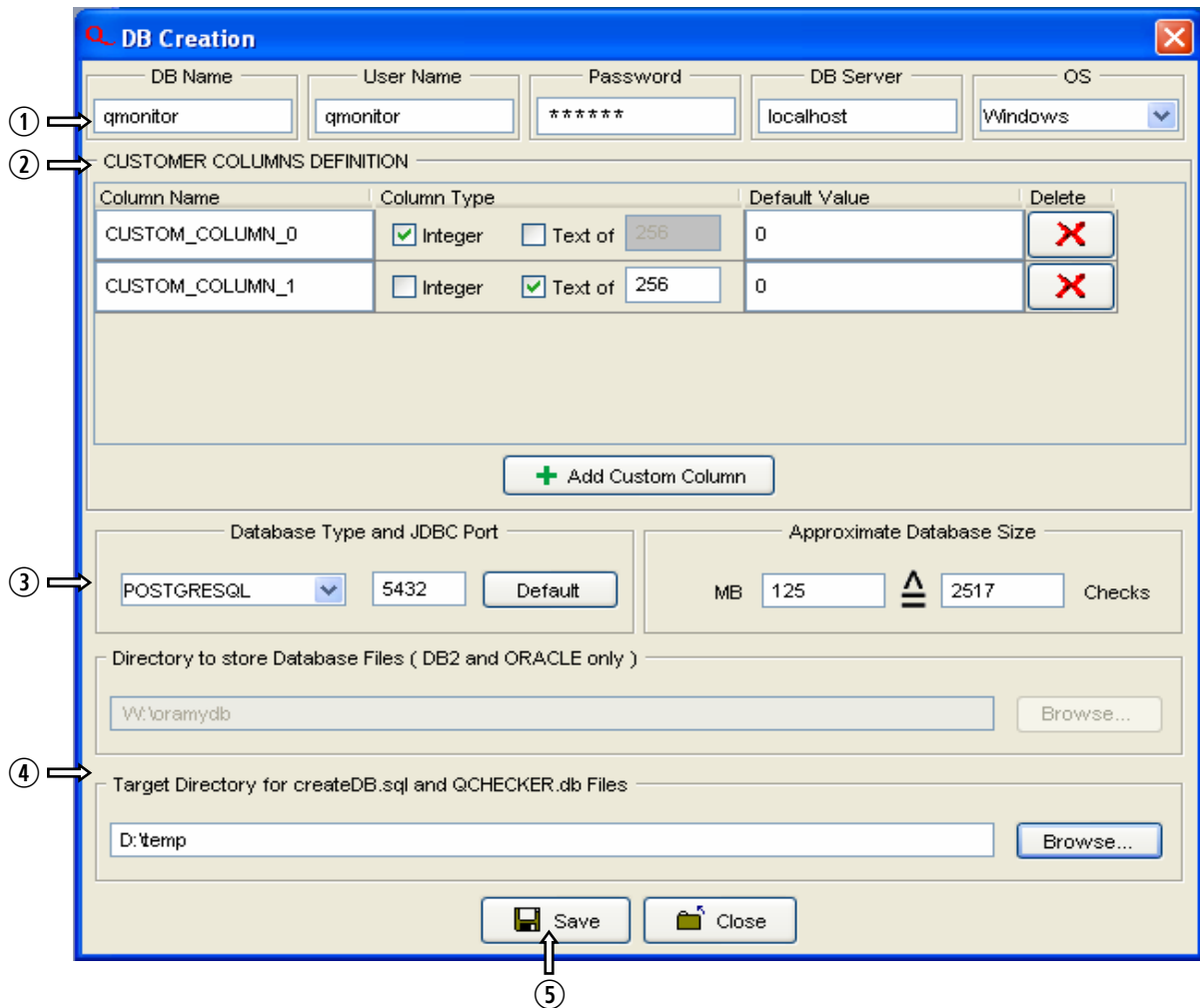


#### STEPS

- (1) Start Q-MONITOR Application
- (2) In the Q-MONITOR menu bar click on the menu item *Database > Database Creation* menu.



→ This will open the following dialog box.



(3) In the DB creation dialog box fill out the database information.

- ①
- DB Name     The name of the database instance that will be created by the SQL script.
  - User Name     The name of the database user that will be able to use this database instance.
  - Password     The user password to open this database instance.
  - DB Server     The name or TCP/IP address of the machine where the DBMS runs.
  - OS            The operating system of the DB-Server.

② Customer Columns Definition

*Optional:* Define customer specific columns name, type and value for the CHECK\_SESSION table.

③ Database type and JDBC port:

Enter database type, JDBC-port number and table size.

Approximative database size:

Specify either the database size (in MB) or the number of checks to be executed. These two boxes are coupled—when entering a value in one of them, the other will be filled automatically.

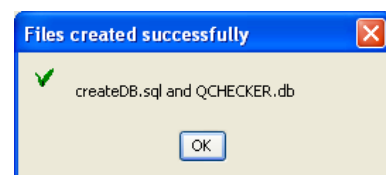
- If specifying the available hard disk space, you will get the number of the checks the results of which can be written on the disk.
- If specifying the number of checks, you will get the disk space required.

In DB2 und ORACLE this specification is used to create tables with a corresponding size. This specification is important only for Oracle and DB2. By all others database management systems instances it is ignored, its use is only for information and administrator support.

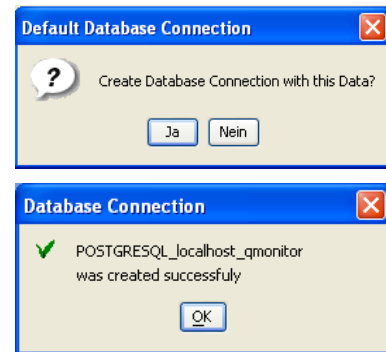
④ Choose a directory where the SQL script `createDB.sql` and the Q-CHECKER database file `QCHECKER.db` are to be created (e.g. `D:\temp`).

(4) Press the *Save* button ⑤.

- This will open the following panel.  
→ Click OK



- The following panel appears.  
→ Click YES to create a new database connection in Q-MONITOR.
- The Database connection is created and set to default in Q-MONITOR.  
→ Click OK



(5) Execute the SQL script on the used database management system (DBMS).

Examples:

- DB2: `DB2 -f createDB.sql`
- ORACLE: `sqlplus adminName/adminPwd @createDB.sql`
- PostgreSQL: `psql -f createDB.sql`
- MySQL: `mysql> source createDB.sql`
- MSSQL: `osql -i createDB.sql`

### 4.2.3 Initial Filling and Updating of Criteria Table

After the initial creation of the database tables the `CRITERION_INFO` table must be filled with data.

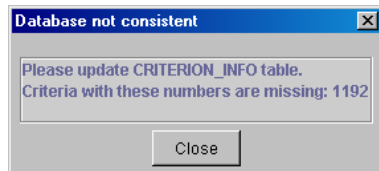
When a newer Q-CHECKER version has been installed on your computer, the criteria numbers in the `CRITERION_INFO` table may differ from the criteria numbers of Q-CHECKER's criteria list. This results in a missing referential integrity between the `CRITERION_NUMBER` entries of the `CHECK_CRITERION` table and the `CRITERION_INFO` table—while Q-CHECKER writes the numbers of the new criteria in the `CHECK_CRITERION` table, in the `CRITERION_INFO` table written by Q-MONITOR this new numbers and the respective criteria names are lacking. If Q-MONITOR queries would be made using this incomplete `CRITERION_INFO` table, the Q-MONITOR reports could be incorrect; at least for the new criteria in the reports no names would appear.

To avoid this problem, the user should periodically execute a check of the criteria table, and, if non-consistency has been detected, update the criteria table—for detailed information see the following text

## Checking the Criteria Table

To check the criteria table, select the menu item *Database > Check Criteria Table*.

If there is detected a inconsistency between the `CRITERION_NUMBER` entries of the `CRITERION_INFO` table and the `CHECK_CRITERION`, a message will be displayed:



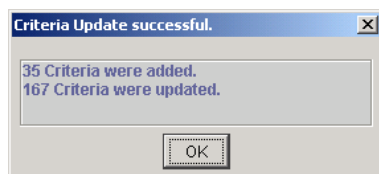
## Updating the Criteria Table

To update the `CRITERION_INFO` table, proceed as follows:

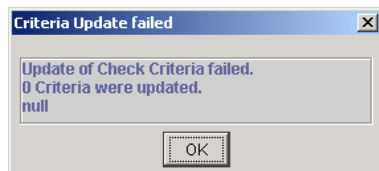


- (1) Select the *Database > Update Criteria Table* menu item.
- (2) In the *Criteria Definition File* dialog box, go to the following directory of your current Q-CHECKER working version:
  - ▶ Q-CHECKER V4: `adminV4/<environment name>/db`
  - ▶ Q-CHECKER V5: `adminV5/<environment name>/db`
 (in Q-CHECKER's initial state go to the directory `adminV5/DEFAULT/db`),
- (3) Select there the `CRITERIA.par` file and press the *Open* button.

The entries in the `CRITERION_INFO` table now will be updated. The update result will be displayed in a message box:



- Message in case of successful update.



- Message in case of error:  
Instead of the `CRITERIA.par` file, a wrong file was selected that contains no criteria definitions.



- Message in case of error:

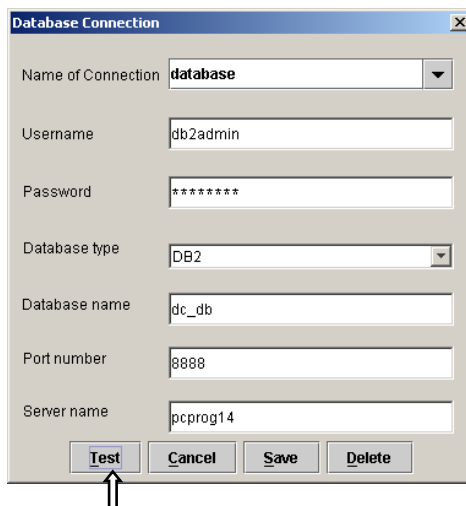
This error can occur when you have upgraded your Q-CHECKER version from a version older than V1.9.1. to V1.9.1 or newer. In this case, the CRITERION\_NAME table must be updated to the new structure (cf. Administrator's Manual—section Database Structure).

To update the database structure, use the DBCREATOR program (for the detailed instruction see Q-MONITOR Installation Manual).

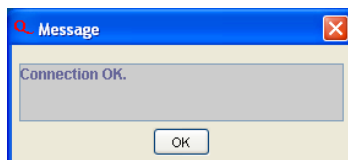
#### 4.2.4 Test Connection with Q-MONITOR



Press the *Database definition* button on the Q-MONITOR toolbar. The *Database Connection* dialog will then open.



Verify in the dialog box whether all specifications are correct. Press then the *Test* button to test the database connection. If the connection is OK, the message below will be displayed.



## 4.3 Administration of Q-CHECKER with JDBC Storing

Prerequisite: JAVA RUNTIME—To write Q-CHECKER check results with Q-CheckerDB in the database, install on each Q-CHECKER workstation JAVA RUNTIME.

### 4.3.1 UNIX/CATIA V4

(1) Adapt QCHECKER.dcls file

```
...
alias TRANSCAT = catia.QCHECKER_LICENSE_DATABASE:STRING;
alias QCLICDB = catia.QCHECKER_LICENSE_DATABASE='YES';
...
```

Set the value of `catia.QCHECKER_LICENSE_DATABASE` on YES to activate the database license.

(2) Adapt QCHECKER.par file.

```
qchecker.DB_CONNECT_BATCH YES
# Write XML report for database connect in interactive run
# (allowed values: YES,NO; default: NO)
qchecker.DB_CONNECT_INTERACTIVE YES
```

Variable name	Explication
DB_CONNECT_BATCH	Set on YES to activating storing for batch.
DB_CONNECT_INTERACTIVE	Set on YES to activating storing for interactive.

(3) Adapt qcheckerV4 script.

```
...
#=====  
# Database definitions  
#=====  
QCHECKER_DB_METHOD="JDBC" # "JDBC" or "FTP"  
#-----  
# JDBC booking  
#-----  
QCHECKER_JAVA_CLASSES="/usr/jdk_base/lib/classes.zip" # Java version 1.1  
QCHECKER_JAVA_CLASSES="/usr/java131/jre/lib/rt.jar" # Java version >1.1  
QCHECKER_JAVA_BOOK="${QCHECKER_LOAD_JAVA}/Q-CheckerDB.jar"  
QCHECKER_JAVA_JDBC="${QCHECKER_LOAD_JAVA}/jdbcdriver.zip"  
QCHECKER_JAVA_CALL="/usr/java131/bin/java"  
...
```

Variable name	Explication
QCHECKER_DB_METHOD	Set the value to JDBC
QCHECKER_JAVA_CLASSES	Path to JAVA base library
QCHECKER_JAVA_JDBC	Path to JDBC driver (driver delivered with database)
QCHECKER_JAVA_CALL	Path to JAVA executable

Copy the JDBC-driver delivered with your database to a common location (e.g.: Q-CHECKER load directory)

Note: With some databases for JAVA 1.1 or JAVA 1.2. different drivers are delivered. If you have JAVA 1.2 (JAVA 2) or higher, use the JAVA 1.2 driver.

#### (4) Adapt QCHECKER.db file



The QCHECKER.db file is not comprised in the Q-MONITOR supply; it must be generated in Q-MONITOR (menu item *Database > Database creation*).

```

Q-Checker configuration file for database connect.
# -----
# Define database type
# (allowed values: DB2,ORACLE)
qchecker.DB_TYPE          DB2
# Define user with write access to database
qchecker.DB_USER         qchecker
# Define password of user with write access to database
qchecker.DB_PASSWORD     qchecker
# Define address of database server
qchecker.DB_SERVER       dbserver
# Define database name
qchecker.DB_NAME         qmon_db
# Define port for remote database access ( for DB2: 8888 ; for ORACLE 1526 )
qchecker.DB_PORT         8888
# -----
# Customer defined database keywords
# -----
...
# qchecker.DB_CUSTOMER_INFO DB_KEY1          TEXT    128  DefaultText
# qchecker.DB_CUSTOMER_INFO DB_KEY2          INTEGER 123
qchecker.DB_CUSTOMER_INFO CHECKING_MODE     TEXT    128  INTERACTIVE
qchecker.DB_CUSTOMER_INFO CUSTOMER_NUMBER   INTEGER 4711

```



For detailed information about the \*DB\_CUSTOMER\_INFO keyword see Q-CHECKER V5 ADMINISTRATION MANUAL—section “Database table rows for user-specific information”.

Variable name	Explication
qchecker.DB_TYPE	database type—possible values: DB2, MSSQL, MYSQL, ORACLE, POSTGRESQL
qchecker.DB_USER	database user
qchecker.DB_PASSWORD	password for database access
qchecker.DB_SERVER	Machine name or IP address of database server
qchecker.DB_NAME	database name
qchecker.DB_PORT	database port for remote database access
qchecker.DB_CUSTOMER_INFO	optional: specify the user defined database columns

### 4.3.2 UNIX/CATIA V5

(1) Adapt CATIA environment file.

```
!-----
!   DASSAULT SYSTEMES - V5 ENVIRONMENT FILE
!-----
! MODE : Global
! TYPE : CATIA
! TMSTMP : 1115806997
!-----
...
QCLICDB=YES
...
```

Set the value of QCLICDB on YES to activate the database license.

(2) Adapt QCHECKER.par file.

```
...
qchecker.DB_CONNECT_BATCH YES
# Write XML report for database connect in interactive run
# (allowed values: YES,NO; default: NO)
qchecker.DB_CONNECT_INTERACTIVE YES
```

Variable name	Explication
qchecker.DB_CONNECT_BATCH	Activate writing in database for batch mode
qchecker.DB_CONNECT_INTERACTIVE	Activate writing in database for interactive mode

## (3) Adapt qcheckerV5 script.

```

...
#=====
# Database definitions
#=====
QCHECKER_DB_METHOD="JDBC"          # "JDBC" or "FTP"
#-----
# JDBC booking
#-----
QCHECKER_JAVA_CLASSES="/usr/jdk_base/lib/classes.zip" # Java version 1.1
QCHECKER_JAVA_CLASSES="/usr/java131/jre/lib/rt.jar" # Java version >1.1
QCHECKER_JAVA_BOOK="{QCHECKER_LOAD_JAVA}/Q-CheckerDB.jar"
QCHECKER_JAVA_JDBC="{QCHECKER_LOAD_JAVA}/jdbcdriver.zip"
QCHECKER_JAVA_CALL="/usr/java131/bin/java"
...

```

Variable name	Explication
QCHECKER_DB_METHOD	Set the value to JDBC
QCHECKER_JAVA_CLASSES	Path to Java base library
QCHECKER_JAVA_JDBC	Path to JDBC driver (driver delivered with database)
QCHECKER_JAVA_CALL	Path to Java executable

Copy the JDBC-driver delivered with your database to a common location (e.g.: Q-CHECKER load directory)

Note: With some databases for JAVA 1.1 or JAVA 1.2. different drivers are delivered. If you have JAVA 1.2 (JAVA 2) or higher, use the JAVA 1.2 driver.

## (4) Adapt QCHECKER.db file.



The QCHECKER.db file is not comprised in the Q-MONITOR supply; it must be generated in Q-MONITOR (menu item *Database > Database creation*).

```

Q-Checker configuration file for database connect.
# -----
# Define database type
# (allowed values: DB2,ORACLE)
qchecker.DB_TYPE          DB2
# Define user with write access to database
qchecker.DB_USER          qchecker
# Define password of user with write access to database
qchecker.DB_PASSWORD      qchecker
# Define address of database server
qchecker.DB_SERVER        dbserver
# Define database name
qchecker.DB_NAME          qmon_db
# Define port for remote database access ( for DB2: 8888 ; for ORACLE 1526 )
qchecker.DB_PORT          8888
# -----
# Customer defined database keywords
# -----
...
# qchecker.DB_CUSTOMER_INFO DB_KEY1          TEXT          128  DefaultText
# qchecker.DB_CUSTOMER_INFO DB_KEY2          INTEGER         123
qchecker.DB_CUSTOMER_INFO  CHECKING_MODE    TEXT            128  INTERACTIVE
qchecker.DB_CUSTOMER_INFO  CUSTOMER_NUMBER  INTEGER         4711

```



For detailed information about the \*DB\_CUSTOMER\_INFO keyword see Q-CHECKER V5 ADMINISTRATION MANUAL—section “Database table rows for user-specific information”.

Variable name	Explication
qchecker.DB_TYPE	database type—possible values: DB2, MSSQL, MYSQL, ORACLE, POSTGRESQL
qchecker.DB_USER	database user
qchecker.DB_PASSWORD	password for database access
qchecker.DB_SERVER	Machine name or IP address of database server
qchecker.DB_NAME	database name
qchecker.DB_PORT	database port for remote database access
qchecker.DB_CUSTOMER_INFO	optional: specify the user defined database columns

### 4.3.3 WINDOWS/CATIA V5

(1) Adapt CATIA environment file.

```
!-----
!   DASSAULT SYSTEMES - V5 ENVIRONMENT FILE
!-----
! MODE : Global
! TYPE : CATIA
! TMSTMP : 1115806997
!-----
...
QCLICDB=YES
...
```

Variable name	Explication
QCLICDB	Set the value of QCLICDB on YES to activate the database license.

(2) Adapt QCHECKER.par file.

```
...
qchecker.DB_CONNECT_BATCH YES
# Write XML report for database connect in interactive run
# (allowed values: YES,NO; default: NO)
qchecker.DB_CONNECT_INTERACTIVE YES
```

Variable name	Explication
qchecker.DB_CONNECT_BATCH	Activate writing in database for batch mode
qchecker.DB_CONNECT_INTERACTIVE	Activate writing in database for interactive mode

(3) Adapt qcheckerV5.vbs script.

```
...
'#####
' # Database definitions
'#####
QCHECKER_DB_METHOD = " JDBC"          '# "JDBC" or "FTP"
'#####
' # JDBC booking
'#####
'#QCHECKER_JAVA_CLASSES = "C:\WINNT\ServicePackFiles\i386\classes.zip"
'# Java version 1.1
QCHECKER_JAVA_CLASSES = "" & "C:\Program Files\Java\j2re1.4.2\lib\rt.jar" & ""
'# Java version >1.1
QCHECKER_JAVA_BOOK = "" & QCHECKER_LOAD_JAVA & "\Q-CheckerDB.jar" & ""
QCHECKER_JAVA_JDBC = "" & QCHECKER_LOAD_JAVA & "\jdbcdriver.zip" & ""
QCHECKER_JAVA_CALL = "" & "C:\Program Files\Java\j2re1.4.2\bin\java.exe" & ""
...
```

Variable name	Explication
QCHECKER_DB_METHOD	Set the value to JDBC
QCHECKER_JAVA_CLASSES	Path to JAVA base library
QCHECKER_JAVA_JDBC	Path to JDBC driver (driver delivered with database)
QCHECKER_JAVA_CALL	Path to JAVA executable

Copy the JDBC-driver delivered with your database to a common location (e.g.: Q-CHECKER load directory)

Note: With some databases for JAVA 1.1 or JAVA 1.2. different drivers are delivered. If you have JAVA 1.2 (JAVA 2) or higher, use the JAVA 1.2 driver.

(4) Adapt QCHECKER.db file.



The QCHECKER.db file is not comprised in the Q-MONITOR supply; it must be generated in Q-MONITOR (menu item *Database > Database creation*).

```

Q-Checker configuration file for database connect.
# -----
# Define database type
# (allowed values: DB2,ORACLE)
qchecker.DB_TYPE          DB2
# Define user with write access to database
qchecker.DB_USER          qchecker
# Define password of user with write access to database
qchecker.DB_PASSWORD      qchecker
# Define address of database server
qchecker.DB_SERVER        dbserver
# Define database name
qchecker.DB_NAME          qmon_db
# Define port for remote database access ( for DB2: 8888 ; for ORACLE 1526 )
qchecker.DB_PORT          8888
# -----
# Customer defined database keywords
# -----
...
# qchecker.DB_CUSTOMER_INFO DB_KEY1          TEXT    128  DefaultText
# qchecker.DB_CUSTOMER_INFO DB_KEY2          INTEGER  123
qchecker.DB_CUSTOMER_INFO  CHECKING_MODE    TEXT    128  INTERACTIVE
qchecker.DB_CUSTOMER_INFO  CUSTOMER_NUMBER  INTEGER  4711

```



For detailed information about the \*DB\_CUSTOMER\_INFO keyword see Q-CHECKER V5 ADMINISTRATION MANUAL—section “Database table rows for user-specific information”.

Variable name	Explication
qchecker.DB_TYPE	database type—possible values: DB2, MSSQL, MYSQL, ORACLE, POSTGRESQL
qchecker.DB_USER	database user
qchecker.DB_PASSWORD	password for database access
qchecker.DB_SERVER	Machine name or IP address of database server
qchecker.DB_NAME	database name
qchecker.DB_PORT	database port for remote database access
qchecker.DB_CUSTOMER_INFO	optional: specify the user defined database columns

## 4.4 Administration of Q-CHECKER with FTP Storing

FTP storing uses a configuration where the Q-CHECKER check result XML files are transferred by FTP to an other location/computer or to a central server. provided for JDBC storing, where the XML files are stored by the Q-CheckerDB program into the Q-MONITOR database.

The advantage of this configuration consists in the following facts:

- As normally on the user computers an FTP program already is installed, there is no need to install there for that purpose any software.
- Users have no access to the database what ensures data security.

Q-CheckerDB is the software provided by TRANSCAT to store Q-CHECKER XML files into the database. It executes the following operations:

- It scans in a predefined time interval a specified directory whether new Q-CHECKER XML files have appeared in it.
- If Q-CHECKER XML files are found, Q-CheckerDB program opens them and stores the Q-CHECKER check-result information into the database.
- Then it moves the Q-CHECKER XML files into a specified destination directory—*either* into a directory where they are conserved, *or* (in case that the database booking was not successful) in an error directory, *or*—in case that as destination directory `/dev/null` has been specified—deletes the Q-CHECKER XML files.
- Writes log files (general protocol file) and (in error case) error protocols.

### 4.4.1 UNIX/CATIA V4

(1) Adapt QCHECKER.dcls file.

```
...
alias TRANSCAT = catia.QCHECKER_LICENSE_DATABASE:STRING;
alias QCLICDB = catia.QCHECKER_LICENSE_DATABASE='YES';
...
```

Variable name	Explication
QCLICDB	Set the value of QCLICDB on YES to activate the database license.

(2) Adapt QCHECKER.par file.

```

...
qchecker.DB_CONNECT_BATCH                YES
# Write XML report for database connect in interactive run
# (allowed values: YES,NO; default: NO)
qchecker.DB_CONNECT_INTERACTIVE         YES
    
```

Variable name	Explication
qchecker.DB_CONNECT_BATCH	Activate writing in database for batch mode
qchecker.DB_CONNECT_INTERACTIVE	Activate writing in database for interactive mode

(3) Adapt qcheckerV4 script.

```

...
#####
# Database definitions
#####
QCHECKER_DB_METHOD="FTP"      # "JDBC" or "FTP"
#-----
...
#-----
# FTP booking
#-----
QCHECKER_FTP_SERVER="dbserver"
QCHECKER_FTP_USER="qchecker"
QCHECKER_FTP_PASSWD="qchecker"
QCHECKER_FTP_DIRECTORY="/data/ftpin"
#####
    
```

Variable name	Explication
QCHECKER.DB_METHOD	Set the value to FTP
QCHECKER_FTP_SERVER	Machine name or IP address of the computer where the Q-CHECKER XML files are transferred to (may be the same machine where the user works, an other computer or a central server)
QCHECKER_FTP_USER	User name
QCHECKER_FTP_PASSWD	Password
QCHECKER_FTP_DIRECTORY	Target directory

(4) Adapt QCHECKER.db file.



The QCHECKER.db file is not comprised in the Q-MONITOR supply; it must be generated in Q-MONITOR (menu item *Database > Database creation*).

```

Q-Checker configuration file for database connect.
# -----
# Define database type
# (allowed values: DB2,ORACLE)
qchecker.DB_TYPE          DB2
# Define user with write access to database
qchecker.DB_USER          qchecker
# Define password of user with write access to database
qchecker.DB_PASSWORD      qchecker
# Define address of database server
qchecker.DB_SERVER        dbserver
# Define database name
qchecker.DB_NAME          qmon_db
# Define port for remote database access ( for DB2: 8888 ; for ORACLE 1526 )
qchecker.DB_PORT          8888
# -----
# Customer defined database keywords
# -----
...
# qchecker.DB_CUSTOMER_INFO DB_KEY1          TEXT      128  DefaultText
# qchecker.DB_CUSTOMER_INFO DB_KEY2          INTEGER  123
qchecker.DB_CUSTOMER_INFO  CHECKING_MODE    TEXT      128  INTERACTIVE
qchecker.DB_CUSTOMER_INFO  CUSTOMER_NUMBER  INTEGER  4711
    
```



For detailed information about the \*DB\_CUSTOMER\_INFO keyword see Q-CHECKER V5 ADMINISTRATION MANUAL—section “Database table rows for user-specific information”.

Variable name	Explication
qchecker.DB_TYPE	database type—possible values: DB2, MSSQL, MYSQL, ORACLE, POSTGRESQL
qchecker.DB_USER	database user
qchecker.DB_PASSWORD	password for database access
qchecker.DB_SERVER	Machine name or IP address of database server
qchecker.DB_NAME	database name
qchecker.DB_PORT	database port for remote database access
qchecker.DB_CUSTOMER_INFO	optional: specify the user defined database columns

## (5) Install and administrate Q-CheckerDB

- Copy the files `Q-CheckerDB.x.x.x.jar` and `Q-CheckerDB_Server.ksh` in a directory.
- Adapt the `Q-CheckerDB_Server.ksh` script to your installation as indicated in the following example file.

## Example file

```
#!/bin/sh
#-----
...
#-----
CLASSPATH="${CLASSPATH}:/data/QCheckerDB/jdbcdriver.zip"
CLASSPATH="${CLASSPATH}:/data/QCheckerDB/Q-CheckerDB.2.1.2.jar"
CLASSPATH="${CLASSPATH}:/usr/java131/jre/lib/rt.jar"
"
/usr/java131/jre/bin/java \
-classpath ${CLASSPATH} \
qcheckerdb.server.QCheckerDB \
-dir /data/ftpin \
-destDir /data/ftpdone \
-errDir /data/ftperror \
-dbType DB2 \
-dbUser qchecker \
-dbPwd qchecker \
-dbName qmon_db \
-dbServer dbserver \
-dbPort 8888 \
-sleep 5000 \
-out /data/QCheckerDB/logs/qcdbserver.log \
-err /data/QCheckerDB/logs/qcdbserver.err
#-----
wait %+
#-----
```

Variable name	Explication
CLASSPATH	Enter class paths to: <ul style="list-style-type: none"> <li>● database JDBC driver</li> <li>● QCheckerDB program</li> <li>● JAVA base library</li> </ul>
Java executable	Path to JAVA executable
-dir	Input directory to be scanned for Q-CHECKER XML files
-destDir	Output directory for processed XML files
-errDir	directory to save XML files the dataset of which could not be saved in database
-dbType	database type—possible values: DB2, MSSQL, MYSQL, ORACLE, POSTGRESQL
-dbUser	database user
-dbPwd	password for database access
-dbName	database name

Variable name	Explication
-dbServer	Machine name or IP address of database server
-dbPort	database port for remote database access
-sleep	interval between two scans (in ms)—default: 5000
-log	directory for general log file
-err	directory for error log file
-errDir	directory to save XML files the dataset of which could not be saved in database

**NOTE:**

- If /dev/null has been specified as destination directory, the XML files after successful booking are deleted.  
If you want to conserve the XML files, save them in an other directory.
- If booking to database failed, the file is moved to the directory specified with the program argument -errDir.
- If the directories specified with the -dir, -destDir, -errDir arguments do not yet exist, then they are created.
- To execute the start of Q-CHECKERDB SERVER as service when starting the computer, do not omit the wait %+ command at the end of the script (if the script is integrated in the /etc/inittab file and respawn is used).

## 4.4.2 UNIX/CATIA V5

(1) Adapt CATIA environment file.

```

!-----
!   DASSAULT SYSTEMES - V5 ENVIRONMENT FILE
!-----
! MODE : Global
! TYPE : CATIA
! TMSTMP : 1115806997
!-----
...
QCLICDB=YES
...

```

Variable name	Explication
QCLICDB	Set the value of QCLICDB on YES to activate the database license.

(2) Adapt QCHECKER.par file.

```

...
qchecker.DB_CONNECT_BATCH                YES
# Write XML report for database connect in interactive run
# (allowed values: YES,NO; default: NO)
qchecker.DB_CONNECT_INTERACTIVE         YES
    
```

Variable name	Explication
qchecker.DB_CONNECT_BATCH	Activate writing in database for batch mode
qchecker.DB_CONNECT_INTERACTIVE	Activate writing in database for interactive mode

(3) Adapt qcheckerV5 script.

```

...
#####
# Database definitions
#####
QCHECKER_DB_METHOD="FTP"      # "JDBC" or "FTP"
#-----
...
#-----
# FTP booking
#-----
QCHECKER_FTP_SERVER="dbserver"
QCHECKER_FTP_USER="qchecker"
QCHECKER_FTP_PASSWD="qchecker"
QCHECKER_FTP_DIRECTORY="/data/ftpin"
#####
    
```

Variable name	Explication
QCHECKER.DB_METHOD	Set the value to FTP
QCHECKER_FTP_SERVER	Machine name or IP address of the computer where the Q-CHECKER XML files are transferred to (may be the same machine where the user works, an other computer or a central server)
QCHECKER_FTP_USER	User name
QCHECKER_FTP_PASSWD	Password
QCHECKER_FTP_DIRECTORY	Target directory

(4) Adapt QCHECKER.db file.



The QCHECKER.db file is not comprised in the Q-MONITOR supply; it must be generated in Q-MONITOR (menu item *Database > Database creation*).

```

Q-Checker configuration file for database connect.
# -----
# Define database type
# (allowed values: DB2,ORACLE)
qchecker.DB_TYPE          DB2
# Define user with write access to database
qchecker.DB_USER          qchecker
# Define password of user with write access to database
qchecker.DB_PASSWORD      qchecker
# Define address of database server
qchecker.DB_SERVER        dbserver
# Define database name
qchecker.DB_NAME          qmon_db
# Define port for remote database access ( for DB2: 8888 ; for ORACLE 1526 )
qchecker.DB_PORT          8888
# -----
# Customer defined database keywords
# -----
...
# qchecker.DB_CUSTOMER_INFO DB_KEY1          TEXT      128  DefaultText
# qchecker.DB_CUSTOMER_INFO DB_KEY2          INTEGER  123
qchecker.DB_CUSTOMER_INFO  CHECKING_MODE     TEXT      128  INTERACTIVE
qchecker.DB_CUSTOMER_INFO  CUSTOMER_NUMBER  INTEGER  4711
    
```



For detailed information about the \*DB\_CUSTOMER\_INFO keyword see Q-CHECKER V5 ADMINISTRATION MANUAL—section “Database table rows for user-specific information”.

Variable name	Explication
qchecker.DB_TYPE	database type—possible values: DB2, MSSQL, MYSQL, ORACLE, POSTGRESQL
qchecker.DB_USER	database user
qchecker.DB_PASSWORD	password for database access
qchecker.DB_SERVER	Machine name or IP address of database server
qchecker.DB_NAME	database name
qchecker.DB_PORT	database port for remote database access
qchecker.DB_CUSTOMER_INFO	optional: specify the user defined database columns

(5) Install and administrate Q-CheckerDB

- Copy the files Q-CheckerDB.x.x.x.jar and Q-CheckerDB\_Server.ksh in a directory.

- Adapt the `Q-CheckerDB_Server.ksh` script to your installation as indicated in the following example file.

### Example file

```
#!/bin/sh
#-----
...
#-----
CLASSPATH="${CLASSPATH}:/data/QCheckerDB/jdbcdriver.zip"
CLASSPATH="${CLASSPATH}:/data/QCheckerDB/Q-CheckerDB.2.1.2.jar"
CLASSPATH="${CLASSPATH}:/usr/java131/jre/lib/rt.jar"
"
/usr/java131/jre/bin/java \
-classpath ${CLASSPATH} \
qcheckerdb.server.QCheckerDB \
-dir /data/ftpin \
-destDir /data/ftpdone \
-errDir /data/ftperror \
-dbType DB2 \
-dbUser qchecker \
-dbPwd qchecker \
-dbName qmon_db \
-dbServer dbserver \
-dbPort 8888 \
-sleep 5000 \
-out /data/QCheckerDB/logs/qcdbserver.log \
-err /data/QCheckerDB/logs/qcdbserver.err
#-----
wait %+
#-----
```

Variable name	Explication
CLASSPATH	Enter class paths to: <ul style="list-style-type: none"> <li>• database JDBC driver</li> <li>• QCheckerDB program</li> <li>• JAVA base library</li> </ul>
Java executable	Path to JAVA executable
-dir	Input directory to be scanned for Q-CHECKER XML files
-destDir	Output directory for processed XML files
-errDir	directory to save XML files the dataset of which could not be saved in database
-dbType	database type—possible values: DB2, MSSQL, MYSQL, ORACLE, POSTGRESQL
-dbUser	database user
-dbPwd	password for database access
-dbName	database name
-dbServer	Machine name or IP address of database server
-dbPort	database port for remote database access
-sleep	interval between two scans (in ms)—default: 5000

Variable name	Explication
-log	directory for general log file
-err	directory for error log file
-errDir	directory to save XML files the dataset of which could not be saved in database



**NOTE:**

- If /dev/null has been specified as destination directory, the XML files after successful booking are deleted. If you want to conserve the XML files, save them in an other directory.
- If booking to database failed, the file is moved to the directory specified with the program argument -errDir.
- If the directories specified with the -dir, -destDir, -errDir arguments do not yet exist, then they are created.

### 4.4.3 WINDOWS/CATIA V5

Adapt the grey marked passages.

(1) Adapt CATIA environment file.

```
!-----
!   DASSAULT SYSTEMES - V5 ENVIRONMENT FILE
!-----
! MODE : Global
! TYPE : CATIA
! TMSTMP : 1115806997
!-----
...
QCLICDB= YES
...
```

Variable name	Explication
QCLICDB	Set the value of QCLICDB on YES to activate the database license.

(2) Adapt QCHECKER.par file.

```
...
qchecker.DB_CONNECT_BATCH YES
# Write XML report for database connect in interactive run
# (allowed values: YES,NO; default: NO)
qchecker.DB_CONNECT_INTERACTIVE YES
```

Variable name	Explication
qchecker.DB_CONNECT_BATCH	Activate writing in database for batch mode
qchecker.DB_CONNECT_INTERACTIVE	Activate writing in database for interactive mode

(3) Adapt qcheckerV5.vbs script.

```
'#=====
'# Database definitions
'#=====
QCHECKER_DB_METHOD = "FTP"      '# "JDBC" or "FTP"
'#-----
...
'#-----
'# FTP booking
'#-----
QCHECKER_FTP_SERVER="dbserver"
QCHECKER_FTP_USER="qchecker"
QCHECKER_FTP_PASSWD="qchecker"
QCHECKER_FTP_DIRECTORY=" C:\Program Files\Transcat PLM\QCheckerDB\data\ftpin"
'#=====
#
```

Variable name	Explication
QCHECKER.DB_METHOD	Set the value to FTP
QCHECKER_FTP_SERVER	Machine name or IP address of the computer where the Q-CHECKER XML files are transferred to (may be the same machine where the user works, an other computer or a central server)
QCHECKER_FTP_USER	User name
QCHECKER_FTP_PASSWD	Password
QCHECKER_FTP_DIRECTORY	Target directory

Remark:

Instead of using FTP transfer, you can also use a simple copy command in the qchecker\_db\_connect () section in qcheckerV5 script.

(4) Adapt QCHECKER.db file.



The QCHECKER.db file is not comprised in the Q-MONITOR supply; it must be generated in Q-MONITOR (menu item *Database > Database creation*).

```

Q-Checker configuration file for database connect.
# -----
# Define database type
# (allowed values: DB2,ORACLE)
qchecker.DB_TYPE          DB2
# Define user with write access to database
qchecker.DB_USER          qchecker
# Define password of user with write access to database
qchecker.DB_PASSWORD      qchecker
# Define address of database server
qchecker.DB_SERVER        dbserver
# Define database name
qchecker.DB_NAME          qmon_db
# Define port for remote database access ( for DB2: 8888 ; for ORACLE 1526 )
qchecker.DB_PORT          8888
# -----
# Customer defined database keywords
# -----
...
# qchecker.DB_CUSTOMER_INFO DB_KEY1          TEXT      128  DefaultText
# qchecker.DB_CUSTOMER_INFO DB_KEY2          INTEGER  123
qchecker.DB_CUSTOMER_INFO  CHECKING_MODE    TEXT      128  INTERACTIVE
qchecker.DB_CUSTOMER_INFO  CUSTOMER_NUMBER  INTEGER  4711
    
```



For detailed information about the \*DB\_CUSTOMER\_INFO keyword see Q-CHECKER V5 ADMINISTRATION MANUAL—section “Database table rows for user-specific information”.

Variable name	Explication
qchecker.DB_TYPE	database type—possible values: DB2, MSSQL, MYSQL, ORACLE, POSTGRESQL
qchecker.DB_USER	database user
qchecker.DB_PASSWORD	password for database access
qchecker.DB_SERVER	Machine name or IP address of database server
qchecker.DB_NAME	database name
qchecker.DB_PORT	database port for remote database access
qchecker.DB_CUSTOMER_INFO	optional: specify the user defined database columns

(5) Install and administrate Q-CheckerDB

- Copy the files Q-CheckerDB.x.x.x.jar and Q-CheckerDB\_Server.vbs in a directory.

- Adapt the Q-CheckerDB\_Server.vbs script to your installation as indicated in the following example file.

### Example file

```

'-----
'#
'#           Q-CheckerDB Server
'#           (C) Transcat PLM GmbH & Co.KG
'#
'#           Am Sandfeld 11c
'#           76149 Karlsruhe
'#           Tel.: +49-721-97043-0
'#
'#
'# start Q-CheckerDB in FTP storing mode
'#
'-----
cmd = "java"
" -classpath "C:\Program Files\Transcat PLM\QCheckerDB\jdbcdriver.zip" & _
" ;"C:\Program Files\Transcat PLM\QCheckerDB\Q-CheckerDB.2.1.2.jar" & _
" qcheckerdb.server.QCheckerDB" & _
" -dir "C:\Program Files\Transcat PLM\QCheckerDB\data\ftpin" & _
" -destDir "C:\Program Files\Transcat PLM\QCheckerDB\data\ftpdone" & _
" -errDir "C:\Program Files\Transcat PLM\QCheckerDB\data\ftperror" & _
" -dbType DB2" & _
" -dbUser qchecker" & _
" -dbPwd qchecker" & _
" -dbName qmon_db" & _
" -dbServer dbserver" & _
" -dbPort 8888" & _
" -sleep 5000" & _
" -out "C:\Program Files\Transcat PLM\QCheckerDB\logs\qcdserver.log" & _
" -err "C:\Program Files\Transcat PLM\QCheckerDB\logs\qcdserver.err" & _
'-----
'MsgBox ":" & cmd & ":"
'-----
Set objShell = wscript.createObject("wscript.shell")
iReturn = objShell.Run( cmd, ,true)
'-----
Wscript.Quit(iReturn)
'-----

```

Variable name	Explication
Java executable	Path to JAVA executable
CLASSPATH	Enter class paths to: <ul style="list-style-type: none"> <li>• database JDBC driver</li> <li>• QCheckerDB program</li> <li>• JAVA base library</li> </ul>
-dir	Input directory to be scanned for Q-CHECKER XML files
-destDir	Output directory for processed XML files
-errDir	directory to save XML files the dataset of which could not be saved in database
-dbType	database type—possible values: DB2, MSSQL, MYSQL, ORACLE, POSTGRESQL
-dbUser	database user
-dbName	database name

Variable name	Explication
-dbPwd	password for database access
-dbServer	machine name or IP address of database server
-dbPort	database port for remote database access
-sleep	interval between two scans (in ms)—default: 5000
-log	directory for general log file
-err	directory for error log file

**NOTE:**

- If /dev/null has been specified as destination directory, the XML files after successful booking are deleted. If you want to conserve the XML files, save them in an other directory.
- If storing in database failed, the file is moved to the directory specified with the program argument -errDir.
- If the directories specified with the -dir, -destDir, -errDir arguments do not yet exist, then they are created by the program.
- To execute Q-CHECKERDB SERVER start as service when starting the computer, you need a service wrapper. (To find one in the Internet, enter in the search engine 'JavaService').

## 4.5 Q-MONITOR Applet

Since version 2.1.1 Q-MONITOR can be run as applet from an internet browser. This allows to start Q-MONITOR from an HTML site.



Figure 4-1: Example: Q-MONITOR start button to start Q-MONITOR from an HTML site (the real appearance depends from the design of the HTML site)

Q-MONITOR will be started when clicking this button. Q-MONITOR in the applet mode has the same functionality as Q-MONITOR has in the native mode—the user can create, save and print out Q-MONITOR reports. The only difference is that the user can not change database connections and save changes in the \*.ini file.

► Advantages of Q-MONITOR applet mode versus native mode:

- Q-MONITOR must be installed and administrated only on the server and not on each client computer where it is used. On client computers, only an internet browser with JAVA plug-ins must be installed.
- The administrator can define which database connections and reports templates can be used.
- User side and administrator side are clearly separated.

To integrate the Q-MONITOR start button in the HTML document, the following code must be embedded:

```
<APPLET
  archive=qmonitor.x.x.x.jar,jdbcdriver.zip
  code=qmon.start.StartQMonitor.class
  height=30 width=130
  align=absMiddle >
</APPLET>
```

This code references Q-MONITOR files and JDBC.jar files that must be findable by the HTTP server.

## Examples

- index.html embedding a frame with the Q-MONITOR start button and referencing the Q-CHECKER side:

```
<HTML>
  <HEAD><TITLE>Q-Checker</TITLE></HEAD>
  <FRAMESET border=0 frameSpacing=0 rows=*,100 frameBorder=NO>
    <FRAME name=mainFrame src="http://www.q-checker.com">
    <FRAME name=bottomFrame src="qserver.html" noResize scrolling=no>
  </FRAMESET>
</HTML>
```

- Example of `qserver.html` containing the Q-MONITOR start button:

```
<HTML><HEAD><TITLE>Q-Monitor</TITLE>
<BODY text=#000000 bgColor=#ffe6be>
  <DIV align=center><FONT size=7>Q-Monitor<BR></FONT>
  <APPLET
    archive=qmonitor.2.5.1.jar,postgresql.jar
    code=qmon.start.StartQMonitor.class
    height=30 width=130 align=absMiddle >
</APPLET>
  </DIV>
</BODY>
</HTML>
```

## 5. Administrating Q-MONITOR

If you have database and SQL skills, you can define new reports that can be used by Q-MONITOR users. To do this, you have to know the structure of the data model that is used by Q-CHECKER to save check data and by Q-MONITOR to evaluate that data. You also have to know how to save the new report definitions in the Q-MONITOR initialization file.

### 5.1 Database Structure

The database in which check results are saved must consist of three tables:

- CHECK\_SESSION
- CHECK\_CRITERION
- CRITERION\_INFO

The CHECK\_SESSION table contains columns for general information from the checks. This information is, for example, model names, model size, date of check, user name, etc. In this table there is only one entry for each check.

In the CHECK\_CRITERION table, information about the criteria checked is saved. This table has as many entries for each check, as there are criteria checked. The exact number depends on the current check profile. From this table you can get information as to whether a criterion was violated or not, how many times it was violated, and what priority it has.

The CRITERION\_INFO table contains the names of the criteria in different languages.

Figure 5-1 shows the three tables, their columns and how they are related.

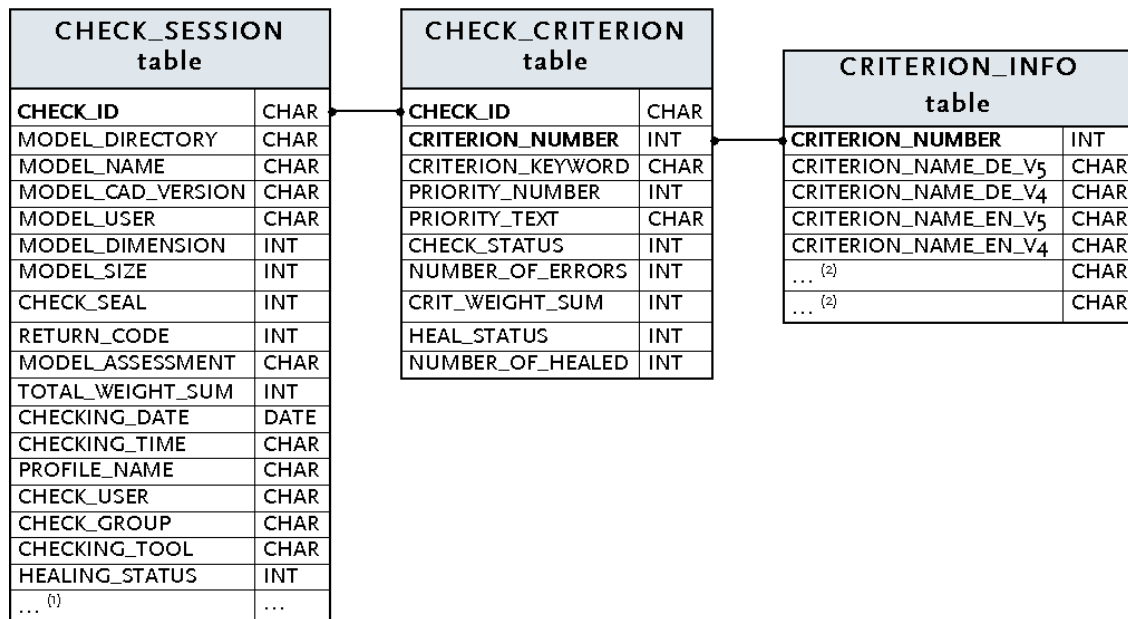


Figure 5-1: Database structure

Explication: (1) There can be additional, user-defined columns.

(2) Depending on the languages defined by the user, there can be additional columns for other languages.

Abbreviations used for data types: CHAR—character, INT—integer

## Restriction Options

The number of restriction options is the same as the number of fields in the database record (i.e. columns in the CHECK\_SESSION table). This means that the test results can be filtered using any of their attributes.

## 5.2 Initialization File \*.ini

Q-MONITOR's initialization file \*.ini contains all configuration settings of Q-MONITOR: the information about database connections, report options and SQL queries used by Q-MONITOR.

The name of this configuration file can be changed by the administrator. The default name is qmonitor.ini. If the administrator does not specify anything else (as program parameter e.g. in a batch file, shell etc.), for the start Q-MONITOR searches for an \*.ini file with the name qmonitor.ini from the current directory.

This information has an XML structure. Certain items are edited automatically when changing the Q-MONITOR settings by the means of the graphic user interface (e.g. database connection data, options) or the wizard (a part of the possible SQL queries). Other items must be edited with a text editor:

- <GRAPH\_XXX>
- <GEN\_FONT\_XXX>
- <GEN\_POS\_MAX>
- <SQL\_TITLE1>
- <TABMOD\_CHECKED>
- <TABMOD\_TYPE>
- <TABMOD\_VALUE1>; TABMOD\_VALUE2>
- a part of the options in the <OPT\_INFO> section
- <GEN\_INFO>
- <HELP\_INFO>

The table below lists the XML tags used in the \*.ini file.

XML tag	Description
<DB_INFO>	<ul style="list-style-type: none"> <li>• Connection info about selectable databases</li> </ul>
<SQL_INFO>	<ul style="list-style-type: none"> <li>• Report definition</li> </ul>
<OPT_INFO>	<ul style="list-style-type: none"> <li>• Options restricting the extent of the query data to be edited in the report</li> </ul>
<GEN_INFO>	<ul style="list-style-type: none"> <li>• General parameters</li> </ul>
<HELP_INFO>	<ul style="list-style-type: none"> <li>• Location of the help file</li> </ul>

The following sections describe each of these tags.

### 5.2.1 DB\_INFO

The section starting with this tag contains parameters needed by the JDBC driver to connect to a database. There are as many DB\_INFO tags as there are database connections defined.

XML tag	Description
<DB_CONN_NAME>	<ul style="list-style-type: none"> <li>Name of the connection</li> </ul>
<DB_TYPE>	<ul style="list-style-type: none"> <li>Database type (DB2, MSSQL, MYSQL, ORACLE, POSTGRESQL)</li> </ul>
<DB_USER>	<ul style="list-style-type: none"> <li>Username for database access</li> </ul>
<DB_PASSWORD>	<ul style="list-style-type: none"> <li>Password for database access</li> </ul>
<DB_NAME>	<ul style="list-style-type: none"> <li>Database name</li> </ul>
<DB_SERVER>	<ul style="list-style-type: none"> <li>Database server (system name / IP address)</li> </ul>
<DB_PORT>	<ul style="list-style-type: none"> <li>Defined JDBC TCP/IP port</li> </ul>

#### Example:

```
<DB_INFO>
  <DB_CONN_NAME>dbserver</DB_CONN_NAME>
  <DB_TYPE>DB2</DB_TYPE>
  <DB_USER>qchecker</DB_USER>
  <DB_PASSWORD>qchecker</DB_PASSWORD>
  <DB_NAME>qmon_db</DB_NAME>
  <DB_SERVER>dbserver</DB_SERVER>
  <DB_PORT>8888</DB_PORT>
</DB_INFO>
```

## 5.2.2 SQL\_INFO

The section starting with the `SQL_INFO` tag contains the report definition information. These are SQL query definition, standard settings for the options window (3D and blocking/numbering options) and for the report chart representation. There are as many `SQL_INFO` tags as there are report templates and reports defined. These are the elements contained in this tag:

XML tag	Description
<SQL_CMD>	<ul style="list-style-type: none"> <li>SQL statement resulting in a 2 dimensional table</li> </ul>
<SQL_HEADER>	<ul style="list-style-type: none"> <li>Header of chart</li> </ul>
<SQL_TITLE1>	<ul style="list-style-type: none"> <li>Title of the 1st column</li> </ul>
<SQL_TITLE2>	<ul style="list-style-type: none"> <li>Title of the 2nd column</li> </ul>
<GRAPH_TYPE>	<ul style="list-style-type: none"> <li>Chart type (bar, pie, ...)</li> </ul>
<GRAPH_3D>	<ul style="list-style-type: none"> <li>3-dimensional view</li> </ul>
<GRAPH_VERTICAL>	<ul style="list-style-type: none"> <li>Vertical/horizontal view</li> </ul>
<GRAPH_ROT_X>	<ul style="list-style-type: none"> <li>1st angle of 3D view</li> </ul>
<GRAPH_ROT_Y>	<ul style="list-style-type: none"> <li>2nd angle of 3D view</li> </ul>
<GRAPH_TAB>	<ul style="list-style-type: none"> <li>Enable/disable chart representation</li> </ul>
<TABMOD_TYPE>	<ul style="list-style-type: none"> <li>Data manipulation type (NUMBERING/BLOCKING). If this tag is missing in the <code>qmonitor.ini</code> file, or if there is a spelling mistake within this tag, the <i>numbering</i> option will be used.</li> </ul>
<TABMOD_VALUE1>	<ul style="list-style-type: none"> <li>1st value</li> </ul>
<TABMOD_VALUE2>	<ul style="list-style-type: none"> <li>2nd value</li> </ul>
<TABMOD_VALUE3>	<ul style="list-style-type: none"> <li>3rd value or YES/NO for percentage</li> </ul>
<3D_CHECKED>	<ul style="list-style-type: none"> <li>3D option checked (YES/NO)</li> </ul>
<3D_COLUMN>	<ul style="list-style-type: none"> <li>Column name for 3rd dimension processing</li> </ul>
<3D_STRING>	<ul style="list-style-type: none"> <li>String value(s) for string columns</li> </ul>
<3D_INTVAL>	<ul style="list-style-type: none"> <li>Integer value(s) for integer columns</li> </ul>
<3D_STARTDATE>	<ul style="list-style-type: none"> <li>Starting date</li> </ul>
<3D_ENDDATE>	<ul style="list-style-type: none"> <li>Ending date</li> </ul>
<3D_INTERVAL>	<ul style="list-style-type: none"> <li>Interval number</li> </ul>
<3D_INTERSTR>	<ul style="list-style-type: none"> <li>Interval string (DAY / MONTH / YEAR)</li> </ul>

**Example:**

```
<SQL_INFO>
  <SQL_CMD>
    SELECT MODEL_ASSESSMENT, COUNT(MODEL_ASSESSMENT)
    AS NUM_OF_MODEL_ASS FROM CHECK_SESSION $#WHERE#$
  </SQL_CMD>
  . . .
</SQL_INFO>
```

**NOTE:**

- A character string delimited by \$#...#\$ in an SQL statement is a placeholder for the WHERE statement.

### 5.2.3 OPT\_INFO

The section starting with this tag defines the properties of the options. Options and their settings are used to restrict the extent of the query data to be edited in the report.

A part of the settings of the options (<OPT\_CHECKED>, <OPT\_VALUE1>, <OPT\_VALUE2>, <OPT\_VALUE3>) can be edited in the options window (if the <OPT\_SHOWN> tag is set on YES).

If in the \*.ini file for an option no section with <OPT\_XXX> tag is written, Q-MONITOR uses the default settings for this option.

XML tags for option settings	Description	Default
<OPT_NAME>	<ul style="list-style-type: none"> <li>Name of option (is taken by Q-MONITOR from the database, defines the column title of the database, should not be changed)</li> </ul>	Column name from the database
<OPT_TITLE>	<ul style="list-style-type: none"> <li>Title text of the option</li> </ul>	Column name from the database
<OPT_SHOWN>	<ul style="list-style-type: none"> <li>Option visible (YES/NO)</li> </ul>	YES
<OPT_CHECKED>	<ul style="list-style-type: none"> <li>Option activated/inactivated (YES/NO)</li> </ul>	NO
<OPT_VALUE1>	<ul style="list-style-type: none"> <li>Value of option field 1 (i.e. a text box, list box etc.)</li> </ul>	–
<OPT_VALUE2>	<ul style="list-style-type: none"> <li>Value of option field 2 (i.e. a text box, list box etc.)</li> </ul>	–
<OPT_VALUE3>	<ul style="list-style-type: none"> <li>Value of option field 3 (i.e. a text box, list box etc.)</li> </ul>	–

## Global and Local Options of Report Templates

Since Q-MONITOR V2.2.1 in the \*.ini file “local” options for report templates can be defined. These “local” options are internal for one only report template. When in a report template a local option is defined, these option overrules the global option.

The use of local options allows to initialize each report template with own individual predefined settings for the visibility, the activation state and the values.



	Global options	Local options
Which of the options?	All options can be used as global and as local options.	
Impact	Command all report templates.	Command only a single report template.
Priority	Taken in account for the individual report template if there is no analogous local option defined.	Take the priority for the individual report template over the global options.
Where defined	in Q-MONITOR's *.ini file within the <OPT_INFO> section	in Q-MONITOR's *.ini file within the <OPT_INFO_LOCAL> section
Editing	Can be edited only in the *.ini file.	Can be edited in the *.ini file and, partially, with the <i>Report Options</i> dialog box..
Tagging	<OPT_INFO> ... </OPT_INFO>	<OPT_INFO_LOCAL> <OPT> ... </OPT> <OPT> ... </OPT> <OPT_INFO_LOCAL>

### Example

Global options definition	Local options definition
<pre> ... &lt;OPT_INFO&gt;   &lt;OPT_TITLE&gt;Healing Status&lt;/OPT_TITLE&gt;   &lt;OPT_NAME&gt;HEALING_STATUS&lt;/OPT_NAME&gt;   &lt;OPT_SHOWN&gt;YES&lt;/OPT_SHOWN&gt; &lt;/OPT_INFO&gt;  &lt;OPT_INFO&gt;   &lt;OPT_NAME&gt;MODEL_CAD_VERSION&lt;/OPT_NAME&gt;   &lt;OPT_SHOWN&gt;YES&lt;/OPT_SHOWN&gt;   &lt;OPT_TITLE&gt;CAD Version (Model)&lt;/OPT_TITLE&gt; &lt;/OPT_INFO&gt; ... </pre>	<pre> ... &lt;SQL_INFO&gt;   &lt;OPT_INFO_LOCAL&gt;     &lt;OPT&gt;       &lt;OPT_TITLE&gt;Model directory&lt;/OPT_TITLE&gt;       &lt;OPT_CHECKED&gt;YES&lt;/OPT_CHECKED&gt;       &lt;OPT_VALUE1&gt;MODELFILE NOT DEFINED&lt;/OPT_VALUE1&gt;       &lt;OPT_NAME&gt;MODEL_DIRECTORY&lt;/OPT_NAME&gt;     &lt;/OPT&gt;     &lt;OPT&gt;       &lt;OPT_TITLE&gt;Healing Status&lt;/OPT_TITLE&gt;       &lt;OPT_SHOWN&gt;NO&lt;/OPT_SHOWN&gt;       &lt;OPT_NAME&gt;HEALING_STATUS&lt;/OPT_NAME&gt;     &lt;/OPT&gt;   ...   &lt;OPT_INFO_LOCAL&gt;   ... </pre>

In the global option section the option with the title *Healing Status* has the status visible (as the <OPT\_SHOWN> setting is *YES*), so that generally the *Healing Status* will be visible in all report templates.

In the local option section the option with the title *Healing Status* has the status not visible (as the <OPT\_SHOWN> setting is *NO*), so that in this individual report template the *Healing Status* option will be hidden (the local option setting overrules for an individual option the global option setting).

## Hiding an active option

The administrator can combine the two settings <OPT\_SHOWN>NO</OPT\_SHOWN> and <OPT\_CHECKED>YES</OPT\_CHECKED>. As a result the option is active, but not visible in the *Report Options* dialog. This state of the option is useful when the administrator wants to prevent the user to deactivate the option or to change its settings.

## 5.2.4 GEN\_INFO

The section starting with this tag defines general Q-MONITOR settings. In the initialization file there is only one GEN\_INFO tag.

XML tag	Description	Default
<GEN_GRAPH_SWITCH>	Maximum allowed number of list elements that can be shown in a chart In this tag the administrator can define a limit number of list elements that can be visualized in charts. Reason: a too big number of list elements might give a unclear visualization in chart form. A new report normally first is opened in chart representation. In the case that the report contains more list elements than defined in the <GEN_GRAPH_SWITCH> tag, the report first will be opened in table form. However the user afterwards can switch the representation to the chart form.	–
<GEN_MAX_OPT_LIST>	Maximum number of entries that can be shown in the list dialog for string options.	–
<GEN_PWD_SAVE>	Saving passwords for database connections in the *.ini file or not.  <ul style="list-style-type: none"> <li>• NO no passwords are saved in the *.ini-file</li> <li>• YES</li> <li>• no &lt;GEN_PWD_SAVE&gt; tag in the *.ini file</li> </ul> } passwords are saved in the *.ini-file	–
<GEN_FONT_NAME>	Name/style/size of a font available in the chart window: used for axis lettering, headers and titles style: PLAIN / BOLD / ITALIC.	–
<GEN_FONT_STYLE>		
<GEN_FONT_SIZE>		
<GEN_HEADER_STYLE>		
<GEN_HEADER_SIZE>		
<GEN_TITLES_STYLE>		
<GEN_TITLES_SIZE>		

XML tag	Description	Default
<GEN_POS_X> <GEN_POS_Y> <GEN_POS_HEGHT> <GEN_POS_WIDTH>	(Since Q-MONITOR Version 2.5.4 ) Position and size of the Q-MONITOR window when opening. Normally, Q-MONITOR memorizes the position of the window, when the user changes its position. So when starting, the window will be in the last position the user has set. These settings take no effect when the setting for <GEN_POS_MAX> is YES. To make active the position and size settings either <GEN_POS_MAX> must be set on NO or the <GEN_POS_MAX> tag must be deleted. When the administrator wants the window be opened in a fixed size and position, he must write the settings in the *.ini file and then write-protect the file.	-
<GEN_POS_MAX>	Opening the Q-MONITOR in maximized size (YES) or not (NO). These setting can be changed only in the *.ini file.	NO
<GEN_GRAPH_COLORS>	(Since Q-MONITOR Version 2.5.4) Colors used for the elements of Q-MONITOR graphs (bars, sectors etc.). If in the *.ini file is no <GEN_GRAPH_COLORS> tag, Q-MONITOR's default colors will be used. The order of the colors in the charts follows the order of the colors in the default list (i.e. the first graph element is green, its neighbor element red and so on).  To define own colors: Each color setting starts with #, followed by a 8- or 6-digit hexadecimal number. 8-digit number: red, green and blue color components (RGB values) and transparency value, 6-digit number: only RGB values (color without transparency)  Colors defined in the *.ini file replace the Q-MONITOR default colors starting with the first one. Example: If in the *.ini file 5 colors are defined, then the first 5 default colors will be replaced by the settings, and the others 7 default colors remain unchanged. The number of color settings is not limited to 12, the user can define as many different colors as he wants.	(1) green (2) red (3) blue (4) orange (5) magenta (6) yellow (7) cyan (8) light-gray (9) white (10) gray (11) dark-gray (12) black

**Example:**

```

<GEN_INFO>
  <GEN_GRAPH_SWITCH>20</GEN_GRAPH_SWITCH>
  <GEN_MAX_OPT_LIST>200</GEN_MAX_OPT_LIST>
  <GEN_FONT_NAME>courier new</GEN_FONT_NAME>
  <GEN_FONT_STYLE>BOLD</GEN_FONT_STYLE>
  <GEN_FONT_SIZE>16</GEN_FONT_SIZE>
  <GEN_POS_X>168</GEN_POS_X>
  <GEN_POS_Y>96</GEN_POS_Y>
  <GEN_POS_HEGHT>645</GEN_POS_HEGHT>
  <GEN_POS_WIDTH>803</GEN_POS_WIDTH>
  <GEN_POS_MAX>NO</GEN_POS_MAX>
  <GEN_HEADER_STYLE>BOLD</GEN_HEADER_STYLE>
  <GEN_HEADER_SIZE>18</GEN_HEADER_SIZE>
  <GEN_PWD_SAVE>YES</GEN_PWD_SAVE>
  <GEN_TITLES_STYLE>BOLD</GEN_TITLES_STYLE>
  <GEN_TITLES_SIZE>16</GEN_TITLES_SIZE>
  <GEN_GRAPH_COLORS>#aa00ff00 #aff0000 #aa0000ff #aaffc800 #aaff00ff #aaffff00 #aa00ffff
                    #aac0c0c0 #ffffff #ff808080 #ff404040 #ff000000
</GEN_GRAPH_COLORS> </GEN_GRAPH_COLORS>
</GEN_INFO>

```

**5.2.5 HELP\_INFO**

The section starting with this tag contains the information about the location of the Q-MONITOR help files.

XML tag	Description
<HELP_EDITOR>	<ul style="list-style-type: none"> <li>Executable file displaying the help document. It is recommended to define it with absolute file path.</li> </ul>
<HELP_FILE>	<ul style="list-style-type: none"> <li>Q-MONITOR help file with path for the native mode</li> </ul>
<HELP_FILE_APPLET>	<ul style="list-style-type: none"> <li>Q-MONITOR help file for the applet mode. Its path must defined relative to the root directory of the HTTP server</li> </ul>

**Example:**

```

<HELP_INFO>
  <HELP_EDITOR>C:\Program files\AcroRd32.exe</HELP_EDITOR>
  <HELP_FILE>J:\qmonitor\Manuals\QMonitor_UserGuide.pdf</HELP_FILE>
  <HELP_FILE_APPLET>/Manuals/QMonitor_UserGuide.pdf</HELP_FILE_APPLET>
</HELP_INFO>

```

## 5.3 Description of Q-CHECKER's XML-Parameters

This section gives a detailed explication of the structure of the data that is written in the database which is analyzed by Q-MONITOR.

### 5.3.1 DB\_INFO Section

Purpose: General data of the database where the check results are recorded by Q-CHECKER.

Keyword	Description	Options	Example
<DB_INFO>	• Start tag of section database parameters from QCHECKER.db		
<DB_TYPE>	Database type	<ul style="list-style-type: none"> <li>• DB2</li> <li>• ORACLE</li> <li>• MYSQL</li> <li>• POSTGRESQL</li> <li>• MSSQL</li> </ul>	DB2
<DB_USER>	Database username	(refer to database documentation)	qchecker
<DB_PASSWORD>	Database password	(refer to database documentation)	qchecker
<DB_SERVER>	Database server address	• IP address	xxx.xxx.xxx.xxx
		• DNS name	qbase.transcat.de
		• as defined in hosts or lmhosts file	qbase
<DB_NAME>	Database name	(refer to database administrator)	qmon_db
<DB_PORT>	Database port for JDBC connections	(refer to database administrator)	8888
</DB_INFO>	• End tag: database parameter section		

### 5.3.2 CHECK\_SESSION Section

Purpose: Global info about check, each check results in one row.

Keyword	Description	Options	Example
<CHECK_SESSION>	• Start tag: global check info section		
<CHECK_ID>	Automatic unique ID, referring to CHECK_CRITERION table		090420031026520 16198
<MODEL_DIRECTORY>	Alias (DDName) name of model file or file tree		DEMO
<MODEL_NAME>	CATIA name of model		QCHECKER DEMO
<MODEL_CAD_VERSION>	Exact CATIA version		CATIA SOLUTIONS V4 RELEASE 2.2 FR 4.2.2
<MODEL_USER>	The 'last-modified' user		catusr1
<MODEL_DIMENSION>	Model dimension as set in the model		10000
<MODEL_SIZE>	Size of the model in Kbyte		817
<CHECK_SEAL>	Result of seal check	<ul style="list-style-type: none"> <li>• 0 = not tested</li> <li>• 1 = OK as defined in profile</li> <li>• 2 = WARNING as def.</li> <li>• 3 = test failed, new check done</li> <li>• 4 = test failed, no check done, assessment changed</li> </ul>	0
<RETURN_CODE>	return code of Q-CHECKER check	<ul style="list-style-type: none"> <li>• 0 = OK, all done</li> <li>• 4 = partially done</li> <li>• 8 = failed</li> </ul>	0
<MODEL_ASSESSMENT>	Assessment text as defined in PROFILE.par file	(refer to PROFILE.par file)	0
<TOTAL_WEIGHT_SUM>	Model assessment points (sum of all CRIT_WEIGHT_SUM)		0
<CHECKING_DATE>	Date of check	• YYYY:MM:DD	2003-04-09
<CHECKING_TIME>	Time of check	• HH:MM:SS	10:26:54
<PROFILE_NAME>	Name of used check profile		USR / test

Keyword	Description	Options	Example
<CHECK_USER>	User, executing the check		catadm
<CHECK_GROUP>	Group of CHECK_USER		catia
<CHECKING_TOOL>	Exact Q-CHECKER version		TransCAT Q-Checker 1.5.3
<HEALING_STATUS>	healing status	<ul style="list-style-type: none"> <li>• 0 = healing not executed</li> <li>• 1 = healing executed</li> </ul>	0
</CHECK_SESSION>	<ul style="list-style-type: none"> <li>• End tag: global check info section</li> </ul>		

### 5.3.3 CHECK\_CRITERION Section

Purpose: Detailed check information–result for each checked criterion, each checked criterion results in one row

Keyword	Description	Options	Example
<CHECK_CRITERION>	<ul style="list-style-type: none"> <li>• Start tag: detailed check information section</li> </ul>		
<CHECK_ID>	Automatic unique ID referring, to CHECK_SESSION table		09042003102652 016198
<CRITERION_NUMBER>	Internal number for criterion, referring to CRITERION_INFO table		1180
<CRITERION_KEYWORD>	Internal keyword for criterion		GeoWireMini Element2D
<PRIORITY_NUMBER>	Priority as defined in PROFILE.par file (assessment of criterion)		1
<PRIORITY_TEXT>	Text for priority from PROFILE.par file		KO Criterion (81)
<CHECK_STATUS>	Status for this criterion	<ul style="list-style-type: none"> <li>• 0 = not called</li> <li>• 1 = OK</li> <li>• 2 = violated</li> <li>• 3 = error</li> </ul>	0
<NUMBER_OF_ERRORS>	Number of faulty elements/criteria (status: Q-CHECKER 1.7.2)		0

Keyword	Description	Options	Example
<CRIT_WEIGHT_SUM>	Weight sum of the criterion, calculated on the base of the criteria weight settings in the PROFILE.par file <ul style="list-style-type: none"> <li>• Geometric elements: <ul style="list-style-type: none"> <li>▶ (number of faulty elements) * (criteria weight)— if <i>Add up weights</i> check box is activated,</li> <li>▶ simple criteria weight—if check box not activated</li> </ul> </li> <li>• Structure elements: simple criteria weight</li> </ul>		0
<HEAL_STATUS>	Status of automatic healing	<ul style="list-style-type: none"> <li>• 0 = not healed</li> <li>• 1 = all healed</li> <li>• 2 = partially healed</li> <li>• 3 = error</li> </ul>	0
<NUMBER_OF_HEALED>	Number of healed elements/criteria (status: Q-CHECKER 1.7.2)		0
</CHECK_CRITERION>	<ul style="list-style-type: none"> <li>• End tag: detailed check information section</li> </ul>		

## 6. UNICODE Enabling

Q-MONITOR can handle UTF-16 encoded strings. Thus the \*.ini file can be a UNICODE document, and UNICODE strings can be displayed on the GUI of Q-MONITOR.

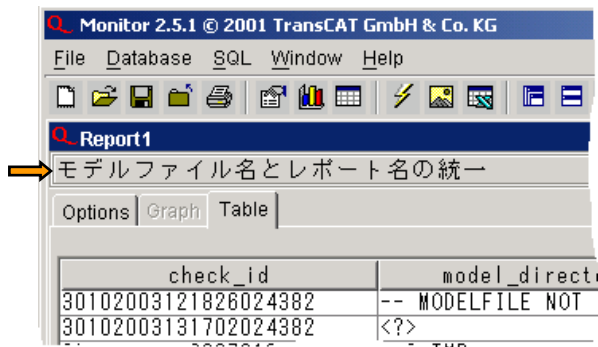


Figure 6-1: Unicode string displayed on the Q-MONITOR GUI

If the Q-MONITOR database contains UNICODE entries (e.g. for criteria or model names), these entries will be also displayed in the Q-MONITOR report graphs and tables.

- To enable Q-MONITOR to display text of a language that requires UNICODE (example for WINDOWS):

a) Write in the \*.ini file in the <GEN\_FONT\_NAME> tag the name of a font, that can depict characters of this language (see file excerpt).

```
...
<GEN_INFO>
  <GEN_FONT_NAME>MS Gothic</GEN_FONT_NAME>
  <GEN_FONT_SIZE>15</GEN_FONT_SIZE>
  <GEN_FONT_STYLE>PLAIN</GEN_FONT_STYLE>
...
```

b) Adapt the batch file qmonitor.bat (the grayed section):

```
@echo off
REM -----
REM Q-Monitor start script
REM -----
set QMONITOR_APPL=".\\qmonitor.3.x.x.jar"
set QMONITOR_JDBC=".\\jdbcdriver.zip"
set JAVAPATH="C:\\Program Files\\Java\\j2re1.4.2\\bin\\java.exe"
set CLASSPATH="C:\\Program Files\\Java\\j2re1.4.2\\lib\\rt.jar;%QMONITOR_APPL%;%QMONITOR_JDBC%"
REM -----
%JAVAPATH% -Dfile.encoding=UTF-16 -classpath %CLASSPATH% qmon.QMonitor .\\qmonitor.ini
REM -----
```

## 7. Trouble Shooting

When Q-MONITOR is run, there is (only) one message output to the command line:

```
qmon.QMonitor.main(String []): Start
```

If more lines are output, then an error has occurred. In the table below some possible errors are listed with their descriptions and remedies.

### ► PROBLEM 1

• Output
Unable to initialize threads: cannot find class <code>java/lang/Thread</code>
• Error description
The JVM cannot find JAVA Runtime Environment classes. This kind of error occurs mostly on UNIX operating systems.
• Solution
Set the correct class path in the start script. The JAVA Runtime Environment classes file is usually named <code>rt.jar</code> and is located in the JAVA installation directory on your system.

**▶ PROBLEM 2****• Output**

```
java.lang.ClassNotFoundException: ...
```

**• Error description**

The JVM cannot find one of the classes it needs.  
Mostly these are JDBC Driver classes.  
The error occurs if invalid paths for \*.jar or \*.zip files are specified  
or if these files are corrupt.

**• Solution**

Check the path settings in the start script. If the settings are correct, check the \*.jar  
and \*.zip files to control be sure whether they are OK.

\*\*\*