TABLE OF CONTENTS

1 INTRODUCTION

This section describes the GLR information model as implemented in SQL server 2000.

The information model describes the design of the tables and their relationships, and forms the foundation on which the GLR RDBMS operates.

2 LOAD DATA

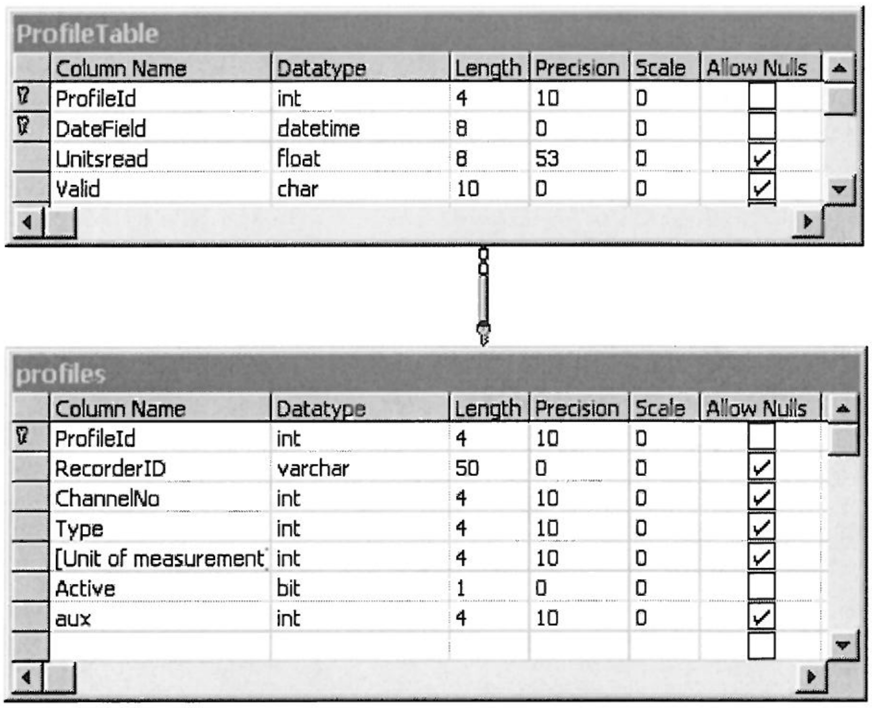
2.1 Tables

Two tables are used to store the load data, *ProfileTable* and *Profiles.*

ProfileTable is where the load data is physically stored. Each reading is time-stamped and associated with a profileid registered in *Profiles*. The measurements are stored in *Unitsread*.

Profiles is where the measured load data are registered and where additional information is stored. Additional information could be information regarding the loggers used to measure the data e.g. the type of data and unit of measurement, e.g. current, kVA, load factor etc.

The structure of the tables is shown below:


ProfileTable

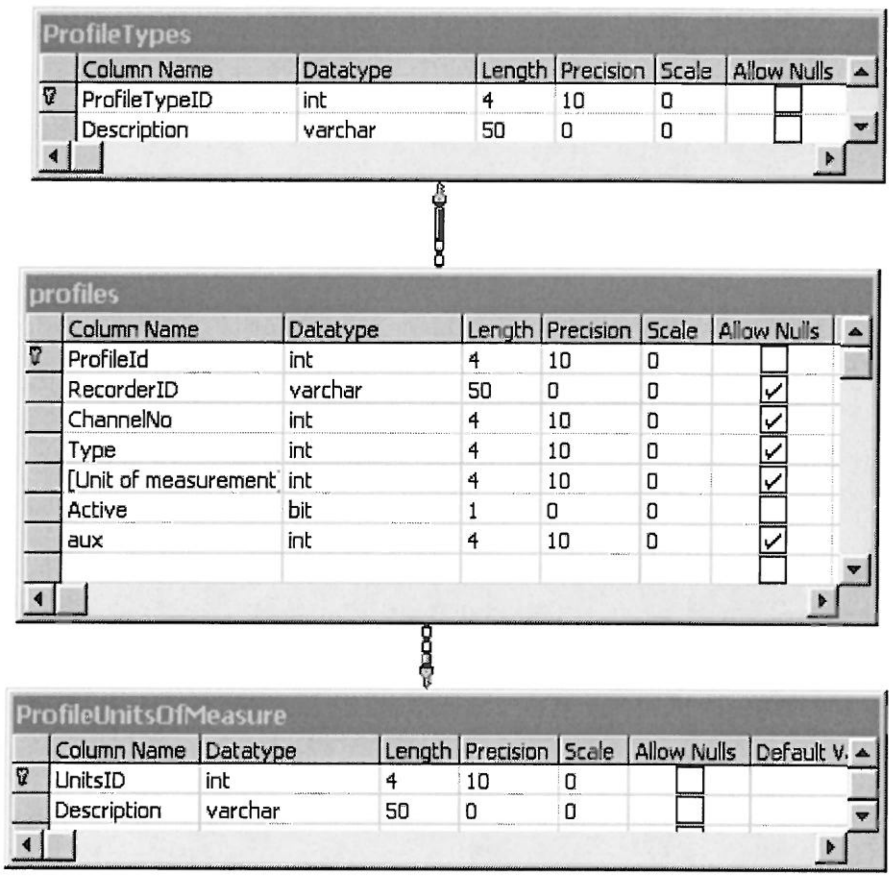
Column Name	Datatype	Length	Precision	Scale	Allow Nulls
ProfileId	int	4	10	0	☐
DateField	datetime	8	0	0	☐
Unitsread	float	8	53	0	✓
Valid	char	10	0	0	✓

profiles

Column Name	Datatype	Length	Precision	Scale	Allow Nulls
ProfileId	int	4	10	0	☐
RecorderID	varchar	50	0	0	✓
ChannelNo	int	4	10	0	✓
Type	int	4	10	0	✓
[Unit of measurement	int	4	10	0	✓
Active	bit	1	0	0	☐
aux	int	4	10	0	✓
					☐

Figure 1 Attributes of ProfileTable and Profiles

The coding for type and unit of measurement is contained in two auxiliary tables:



ProfileTypes

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	ProfileTypeID	int	4	10	0	☐
	Description	varchar	50	0	0	☐

profiles

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	ProfileId	int	4	10	0	☐
	RecorderID	varchar	50	0	0	✓
	ChannelNo	int	4	10	0	✓
	Type	int	4	10	0	✓
	[Unit of measurement	int	4	10	0	✓
	Active	bit	1	0	0	☐
	aux	int	4	10	0	✓
						☐

ProfileUnitsOfMeasure

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls	Default V.
🔑	UnitsID	int	4	10	0	☐	
	Description	varchar	50	0	0	☐	

Figure 2 The coding for "type" and "unit of measurement" is contained in two auxiliary tables

2.2 Relationships

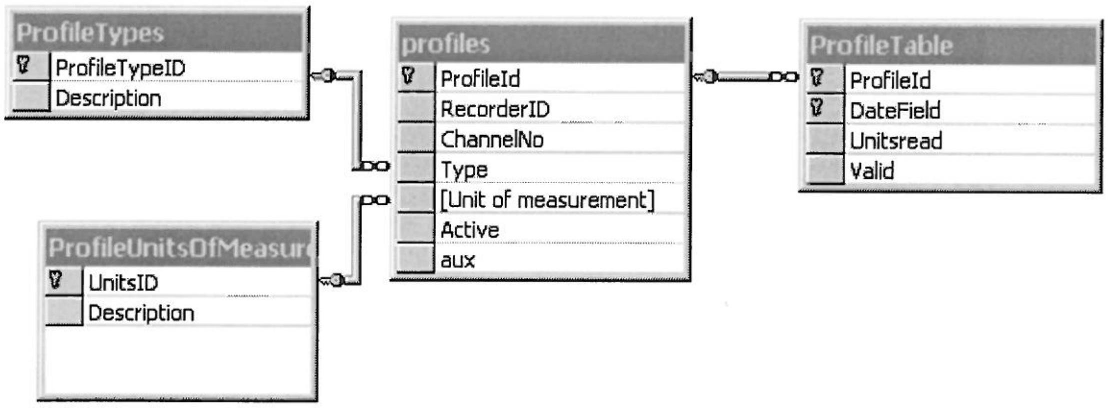


Figure 3 Tables for storing load information

A foreign key relationship exists between *Profiles* and *ProfileTable* on ProfileId.

A foreign key relationship exists between *ProfileTypes* and *Profiles* on ProfileTypeID.
A foreign key relationship exists between *ProfileUnitsOfMeasure* and *Profiles* on UnitsID.

3 CONSUMER RESPONSES

3.1 Tables

Four tables are used to store the consumer responses:

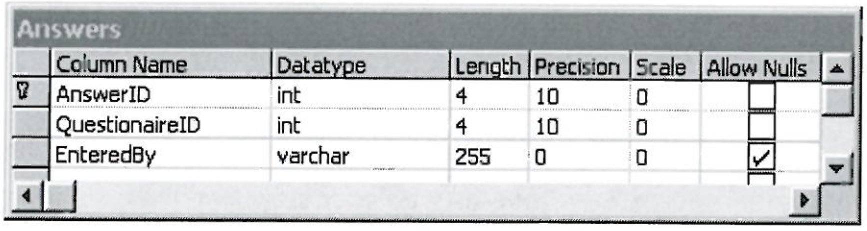
- *Answers*
- *Answers_blob*
- *Answers_character*
- *Answers_number*

Answers is a register of the answers and is essentially a list of valid AnswerID. It also establishes a link with a questionnaire through the QuestionnaireID.

The answers are stored in one of the other three tables depending on the data type. Three data types are supported: blob, character and number.

The data type and the mapping between QuestionID and ColumnNumber are defined in *Questions* (see section 4 for more details).

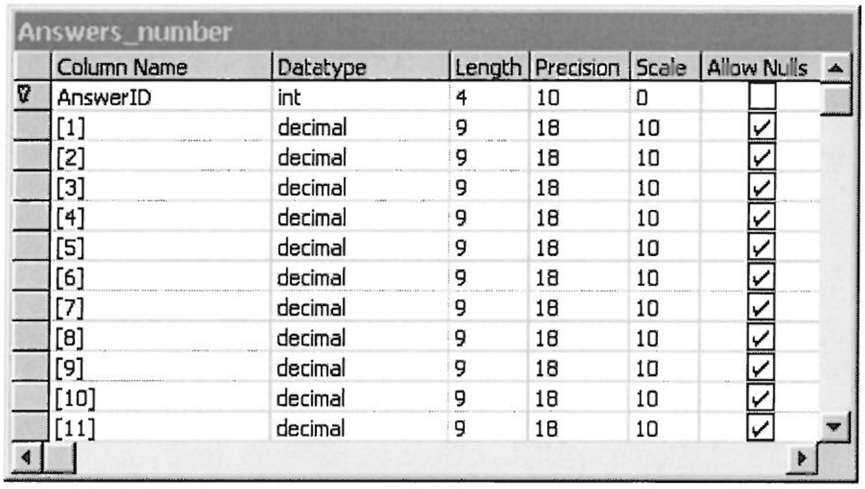
The structure of the tables is shown below:



Answers

Column Name	Datatype	Length	Precision	Scale	Allow Nulls
AnswerID	int	4	10	0	☐
QuestionaireID	int	4	10	0	☐
EnteredBy	varchar	255	0	0	✓

Figure 4 Attributes of *Answers*



Answers_number

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	AnswerID	int	4	10	0	☐
	[1]	decimal	9	18	10	✓
	[2]	decimal	9	18	10	✓
	[3]	decimal	9	18	10	✓
	[4]	decimal	9	18	10	✓
	[5]	decimal	9	18	10	✓
	[6]	decimal	9	18	10	✓
	[7]	decimal	9	18	10	✓
	[8]	decimal	9	18	10	✓
	[9]	decimal	9	18	10	✓
	[10]	decimal	9	18	10	✓
	[11]	decimal	9	18	10	✓

Figure 5 Attributes of *Answers_datatype*

Note that the same structure is used for the different data types. Columns [1],[2] etc. are typed based on the data type of the table, e.g. if the table was named Answers_character, the columns would be of type character.

3.2 Relationships

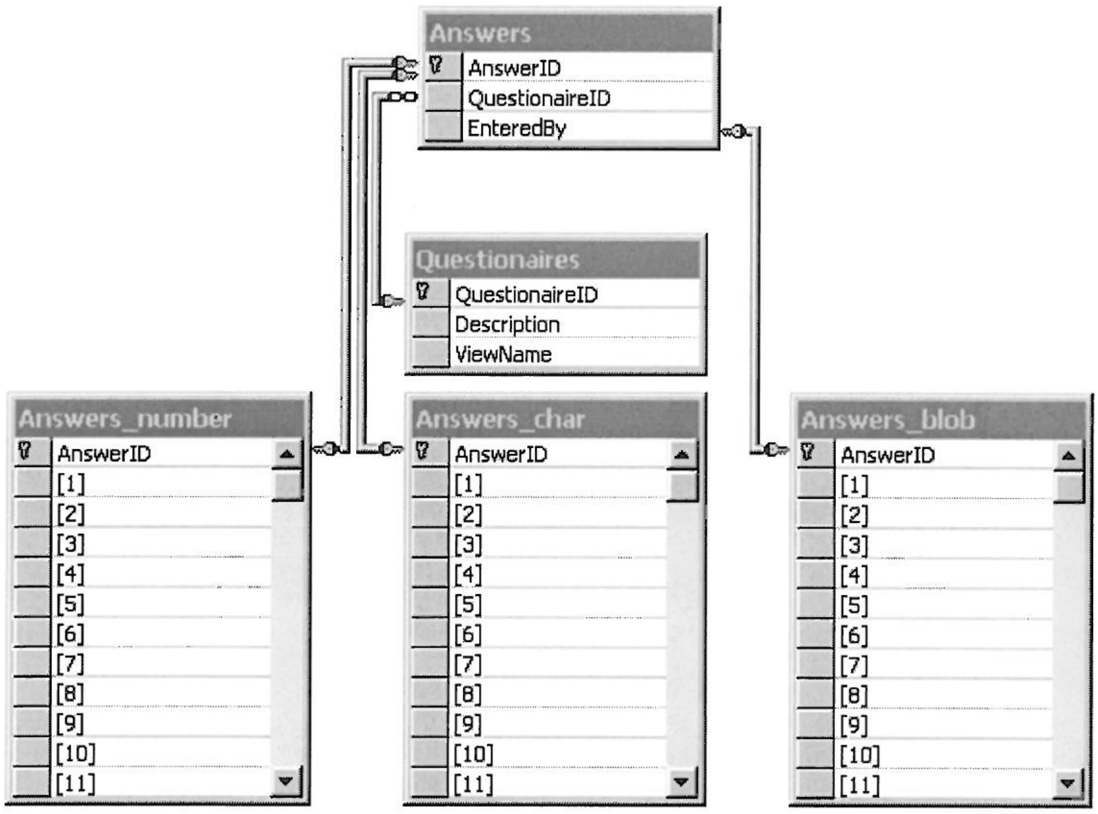


Figure 6 Tables for storing consumer response information

A foreign key relationship exists between *Answers* and *Answers_blob, Answers_character, Answers_number* on AnswerId.

A foreign key relationship also exists between *Questionaires* and *Answers_blob, Answers_character, Answers_number* on QuestionaireId (see section 4).

4 QUESTIONS

4.1 Tables

Two tables are used to store the questions:

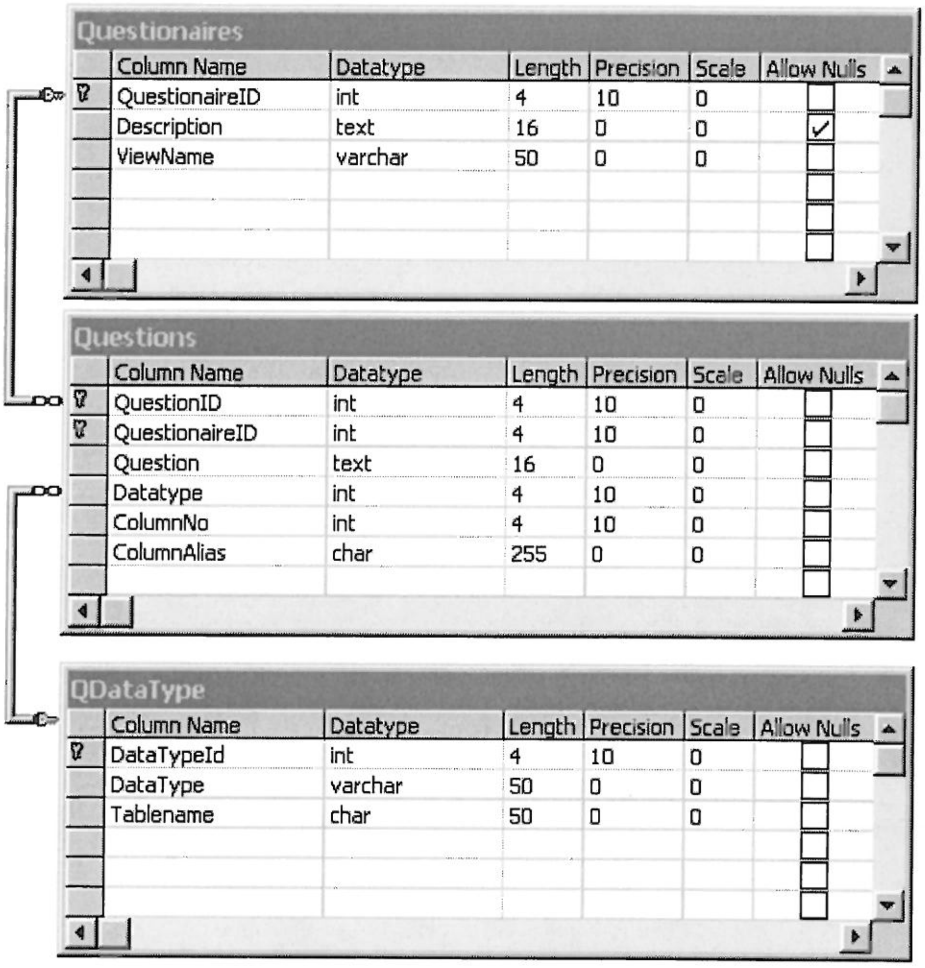
- *Quesions*
- *Questionaires*

Questionaires is a register of valid QuestionaireID's.

Questions contains the following fields:

- Question - the question in words
- Datatype – the data type of the expected answer. This determines in which table the answer is stored (see section 3)
- ColumnNo – the column name where the answer to this question will be stored in the answer table.

The structure of the tables is shown below:



Questionaires

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	QuestionaireID	int	4	10	0	☐
	Description	text	16	0	0	✓
	ViewName	varchar	50	0	0	☐

Questions

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	QuestionID	int	4	10	0	☐
🔑	QuestionaireID	int	4	10	0	☐
	Question	text	16	0	0	☐
	Datatype	int	4	10	0	☐
	ColumnNo	int	4	10	0	☐
	ColumnAlias	char	255	0	0	☐

QDataType

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	DataTypeId	int	4	10	0	☐
	DataType	varchar	50	0	0	☐
	Tablename	char	50	0	0	☐

Figure 7 Attributes of the Questionaires and Questions tables

A further UNIQUE constraint is placed on Datatype and ColumnNo, to ensure unique referencing between the questions and the three answer tables.

The encoding for Datatype is contained in the Qdatatype table

4.2 Relationships

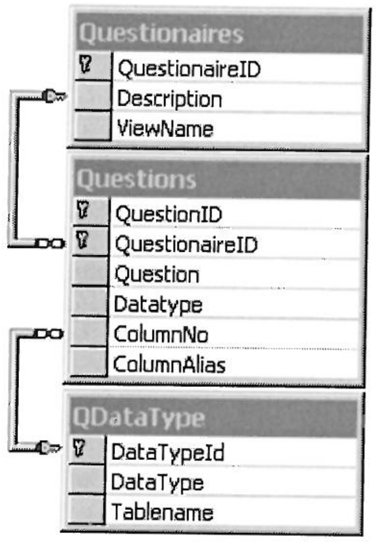


Figure 8 Tables for storing question information

A foreign key relationship exists between *Questionaires* and *Questions* on QuestionaireId.
A foreign key relationship exists between *Questionaires* and *QDatatype* on QDatatypeId.

5 GROUPING

5.1 Tables

Two tables are used to store the group information:

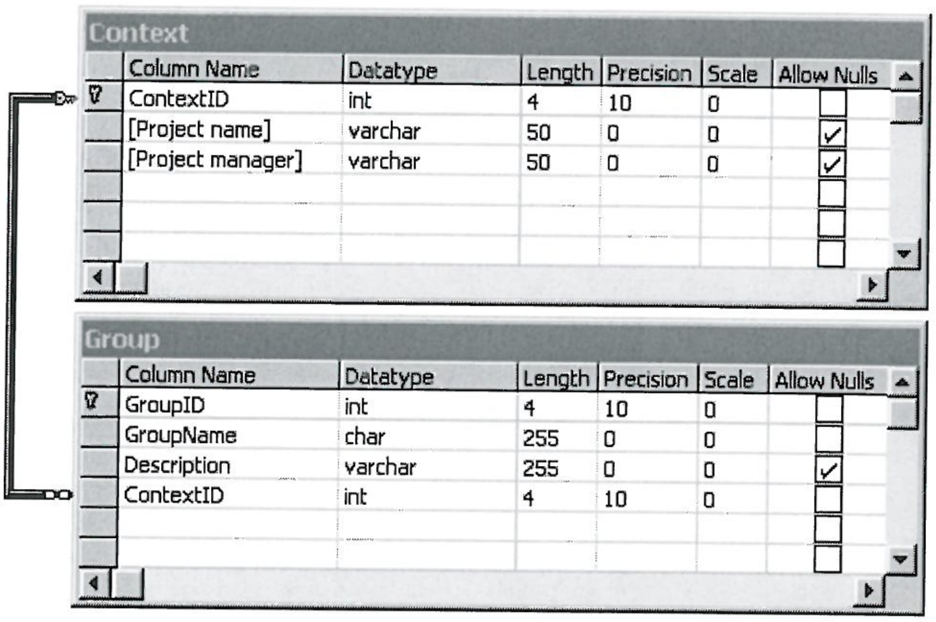
- *Group*
- *Context*

Context is a register of valid ContextID's.

Group contains the following fields:

- GroupID – Primary key for each group
- GroupName – A hierarchal name for each group. The group name is created as follows: Parent|Parent|Parent|Child, e.g. NRSProject|Helderberg|1998. This structure enables the user to work with either a specific group of consumers or a group and all its children.
- ContextID – The context in which the answers and load data is stored (see section 5).

The structure of the tables is shown below:



Context

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	ContextID	int	4	10	0	☐
	[Project name]	varchar	50	0	0	✓
	[Project manager]	varchar	50	0	0	✓

Group

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	GroupID	int	4	10	0	☐
	GroupName	char	255	0	0	☐
	Description	varchar	255	0	0	✓
	ContextID	int	4	10	0	☐

Figure 9 Attributes of the Context and Group tables

5.2 *Relationships*

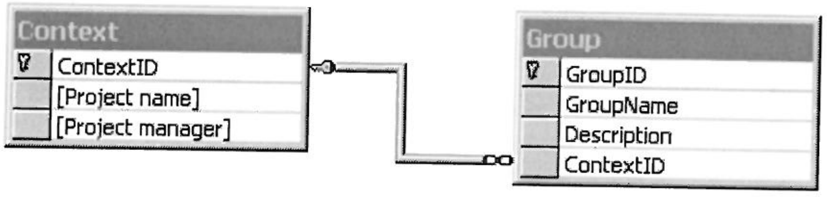


Figure 10 Tables for storing context and grouping information

A foreign key relationship exists between *Context* and *Group* on ContextId.

6 DMO (DATA MANIPULATION OBJECTS)

6.1 Tables

Two tables are used to store the DMO information:

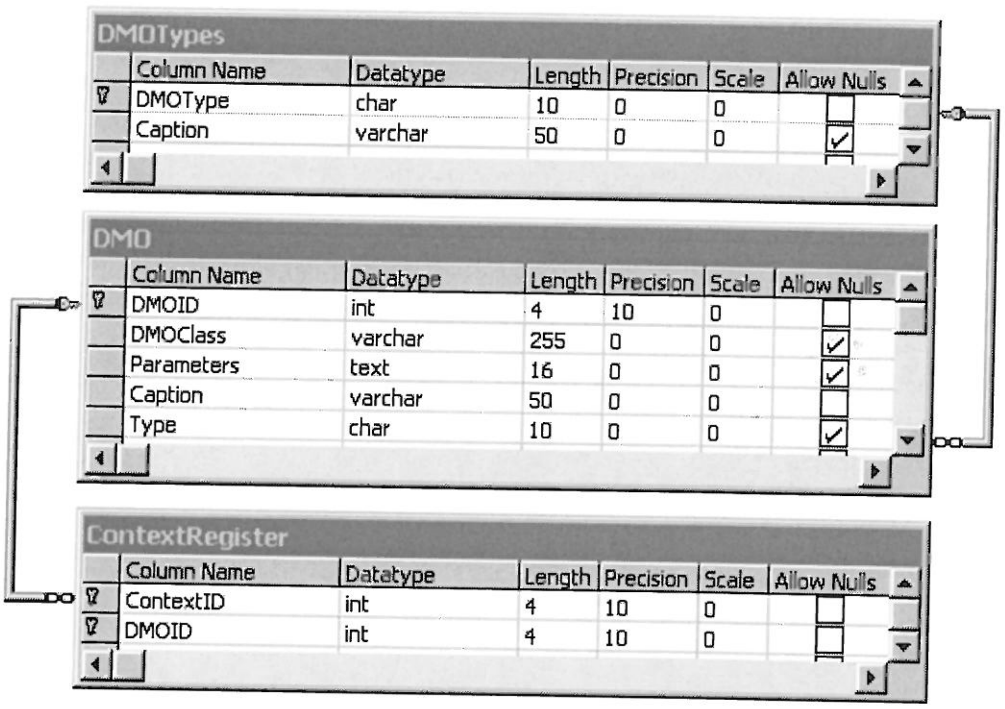
- *DMO*
- *ContextRegister*

DMO is a register of valid DMOID's. It also contains the class name for a specific DMO which is the executable filename for a specific module. The Parameters column is a *text* type field and contains setup information for the module. The Caption is a short description of the module.

The type column is used to group the modules and a list of valid types is contained in

ContextRegister establishes a many to many link between *Context* and *DMO*. This table defines which DMO will be used with which context.

The structure of the tables is shown below:


DMOTypes

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	DMOType	char	10	0	0	☐
	Caption	varchar	50	0	0	✓

DMO

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	DMOID	int	4	10	0	☐
	DMOClass	varchar	255	0	0	✓
	Parameters	text	16	0	0	✓
	Caption	varchar	50	0	0	☐
	Type	char	10	0	0	✓

ContextRegister

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	ContextID	int	4	10	0	☐
🔑	DMOID	int	4	10	0	☐

Figure 11 **Attributes of the DMO and ContextRegister tables**

6.2 Relationships

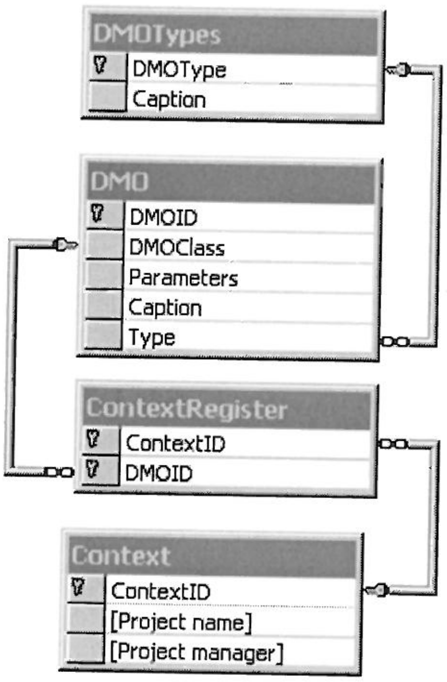


Figure 12 **Tables for storing DMO information**

A foreign key relationship exists between *DMO* and *ContextRegister* on DMOID. Another foreign key relationship exists between *ContextRegister* and *Context* on ContextID. This creates a many-to-many relationship between *Context* and *DMO.*

A foreign key relationship exists between *DMO* and *DMOTypes* on DMOType.

7 LINKING

7.1 *Tables*

Two tables are used to store the linking information:

- *Linktable*
- *Consumer*

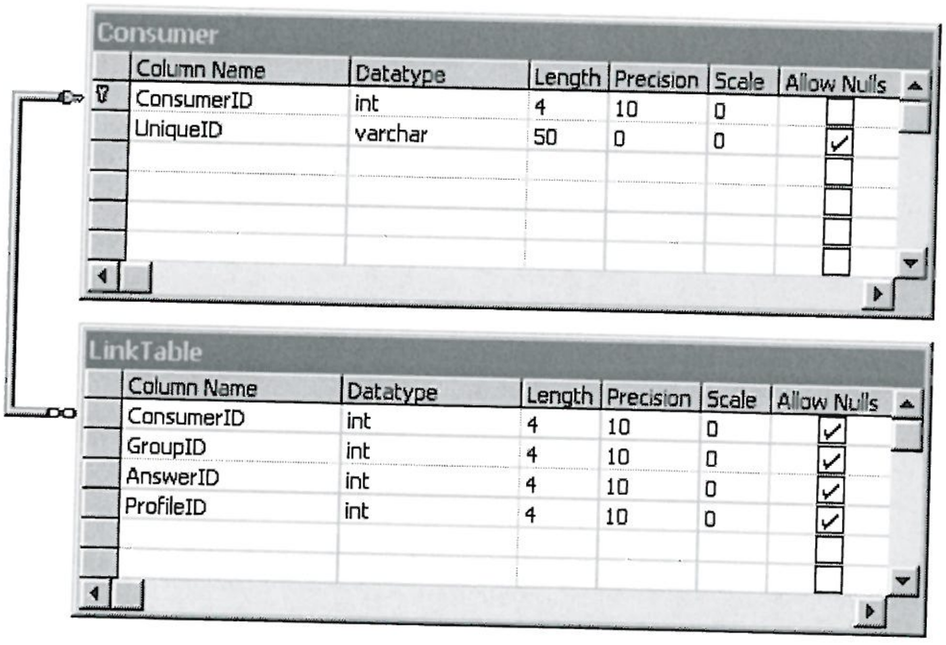
Consumer is a register of valid ConsumerID's.

Linktable contains the following fields:

- ConsumerID – a valid consumer id
- GroupID – a valid group id (see section 5)
- ProfileID – a valid profile id (see section 2)
- AnswerID – a valid answer id (see section 3)

A trigger with the following SQL is associated with linktable to delete any invalid entries: *"delete from linktable where answerid=0 and profileid=0 and consumerid=0"*

The structure of the tables is shown below:



Consumer

Column Name	Datatype	Length	Precision	Scale	Allow Nulls
ConsumerID	int	4	10	0	
UniqueID	varchar	50	0	0	✓

LinkTable

Column Name	Datatype	Length	Precision	Scale	Allow Nulls
ConsumerID	int	4	10	0	✓
GroupID	int	4	10	0	✓
AnswerID	int	4	10	0	✓
ProfileID	int	4	10	0	✓

Figure 13 Attributes of the Consumer and LinkTable tables

The UniqueID in the *Consumer* table is an external unique identification for the ID. This might be GPS, meter number etc.

7.2 Relationships

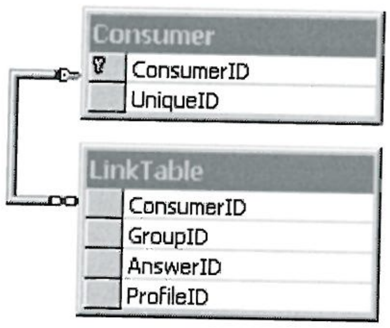


Figure 14 Tables for storing linking information

A foreign key relationships exist between *LinkTable* and the followings tables:

- Answers on AnswerID
- Consumer on ConsumerID
- Group on GroupID
- Profile on ProfileID

8 CONSUMER RESPONSE QUALITY RULES

The rules contained in these tables are used by the SocioChecker.Exe module.

Three types of rules can be stored:

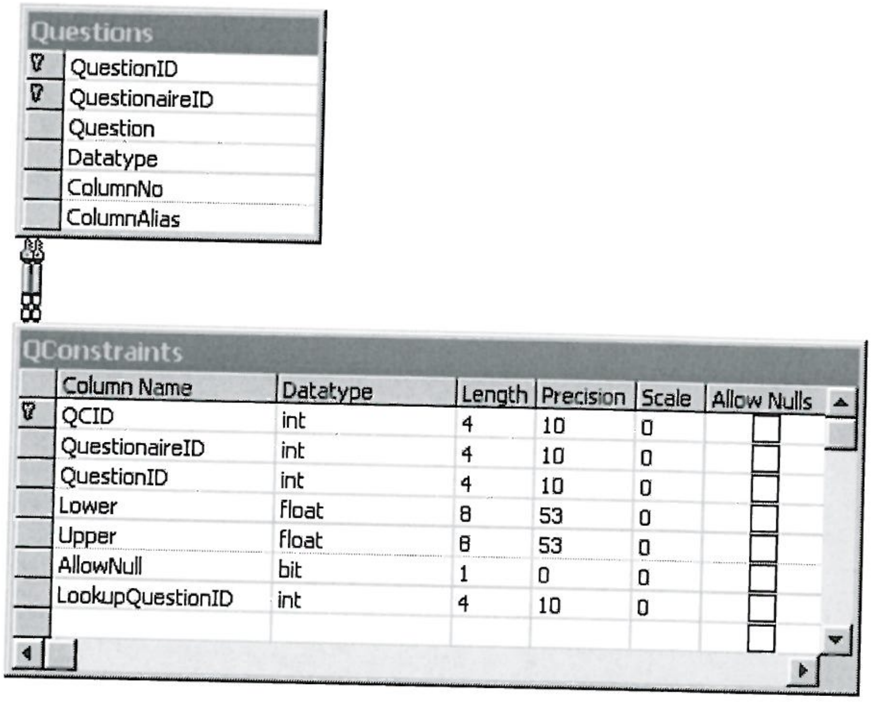
- Domain
- Redundancy
- Model

For more information see the user guide of SocioChecker.Exe.

8.1 Domain

The domain tests are stored in a single table called Qcontraints. QCID is a unique identifier for each constraint. The upper and lower bounds for numerical values are stored in *Lower* and *Upper.* The *AllowNull* field indicates where a null value for the question is allowed.

The structure of the tables is shown below:



Questions

🔑	QuestionID
🔑	QuestionaireID
	Question
	Datatype
	ColumnNo
	ColumnAlias

QConstraints

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	QCID	int	4	10	0	☐
	QuestionaireID	int	4	10	0	☐
	QuestionID	int	4	10	0	☐
	Lower	float	8	53	0	☐
	Upper	float	8	53	0	☐
	AllowNull	bit	1	0	0	☐
	LookupQuestionID	int	4	10	0	☐

Figure 15 Attributes of the QConstrains tables

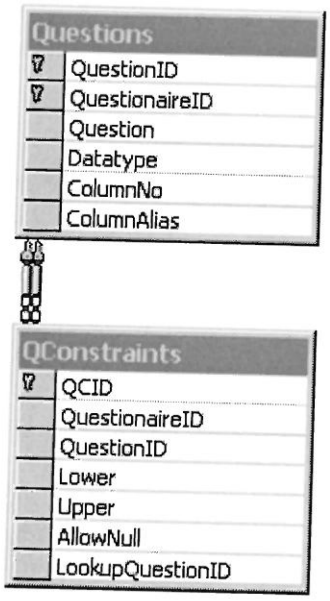


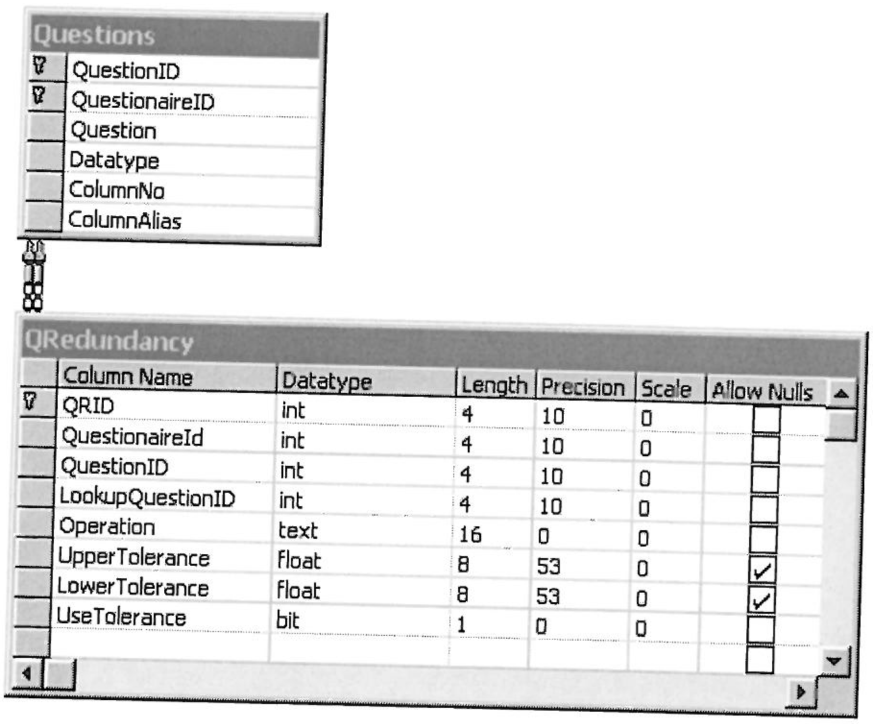
Figure 16 Tables for storing domain type rules

A foreign key relationship exist between *Questions* and *Qconstraints* on QuestionID and QuestionaireID.

8.2 Redundancy

The redundancy tests are stored in a single table called QRedundancy. *QRID* is a unique identifier for each constraint. The upper and lower bounds of the result of the operation specified in *Operation* are stored in *Lower Tolerance* and *Upper Tolerance.* If *UseTolerance* field is false, the test fails if the result of the operation, specified in *Operation,* is not zero.

The structure of the tables is shown below:



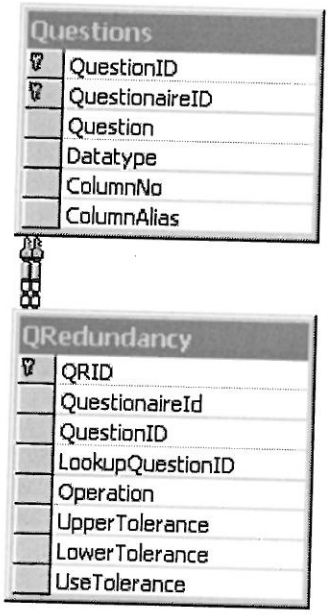
Questions

🔑	QuestionID
🔑	QuestionaireID
	Question
	Datatype
	ColumnNo
	ColumnAlias

QRedundancy

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	QRID	int	4	10	0	☐
	QuestionaireId	int	4	10	0	☐
	QuestionID	int	4	10	0	☐
	LookupQuestionID	int	4	10	0	☐
	Operation	text	16	0	0	☐
	UpperTolerance	float	8	53	0	✓
	LowerTolerance	float	8	53	0	✓
	UseTolerance	bit	1	0	0	☐

Figure 17 Attributes of the QRedundancy table



Questions

🔑	QuestionID
🔑	QuestionaireID
	Question
	Datatype
	ColumnNo
	ColumnAlias

QRedundancy

🔑	QRID
	QuestionaireId
	QuestionID
	LookupQuestionID
	Operation
	UpperTolerance
	LowerTolerance
	UseTolerance

Figure 18 Tables for storing redundancy type rules

A foreign key relationship exist between *Questions* and *QRedundancy* on QuestionID and QuestionaireID.

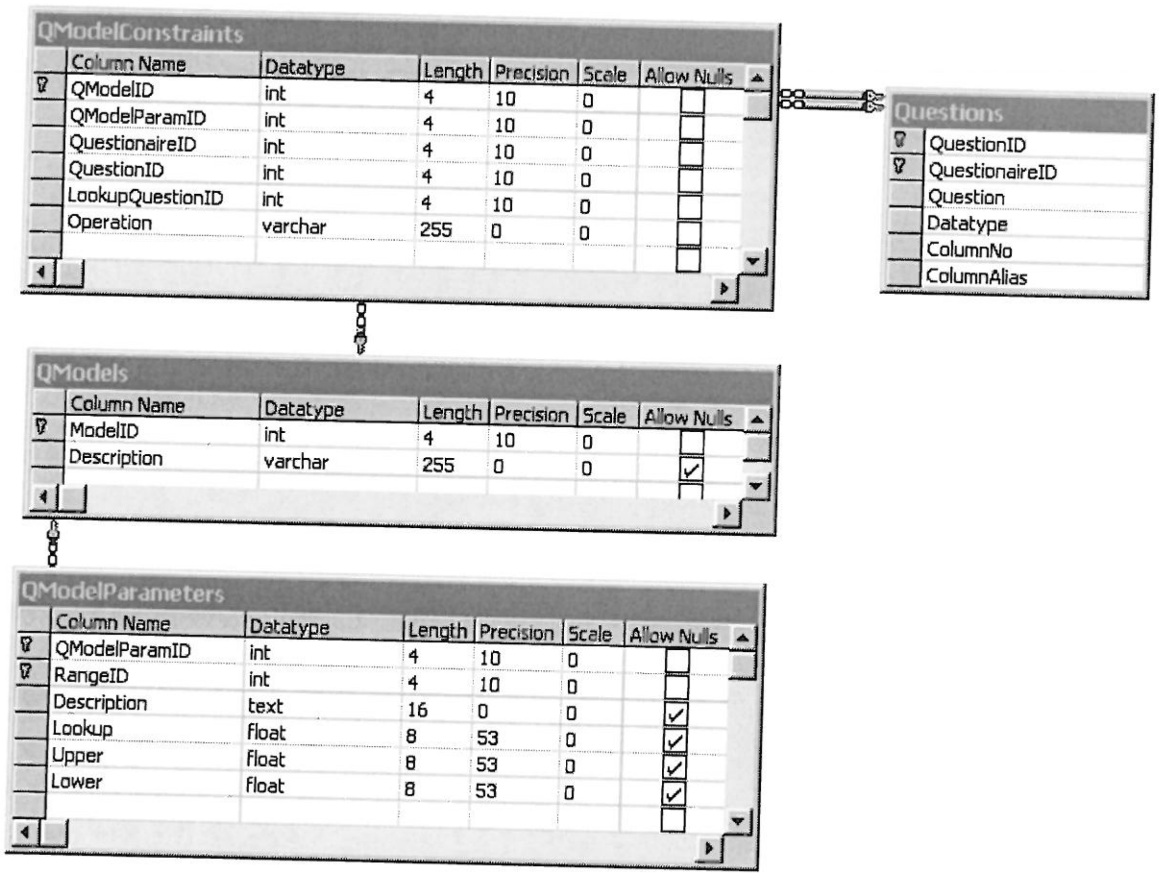
8.3 *Model*

The model type tests are stored in three tables

- Qmodels
- QmodelParameters
- QmodelConstraints

Qmodels contains a list of valid ModelIDs and a description of the model. QmodelParameters stores the parameters for the model. QmodelConstraints stores the *operation* to be performed on the consumer response.

A model test fails if the value of the consumer response is larger than *upper* or smaller than *lower* for a specific *lookup*. The value in *lookup* is compared against the result of the *operation* to determine which set of *upper* and *lower* bounds to test for.



QModelConstraints

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	QModelID	int	4	10	0	☐
	QModelParamID	int	4	10	0	☐
	QuestionaireID	int	4	10	0	☐
	QuestionID	int	4	10	0	☐
	LookupQuestionID	int	4	10	0	☐
	Operation	varchar	255	0	0	☐

Questions

🔑	QuestionID
🔑	QuestionaireID
	Question
	Datatype
	ColumnNo
	ColumnAlias

QModels

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	ModelID	int	4	10	0	☐
	Description	varchar	255	0	0	✓

QModelParameters

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	QModelParamID	int	4	10	0	☐
🔑	RangeID	int	4	10	0	☐
	Description	text	16	0	0	✓
	Lookup	float	8	53	0	✓
	Upper	float	8	53	0	✓
	Lower	float	8	53	0	✓

Figure 19 **Attributes of the Qmodels, QmodelConstraints and Qmodelparameter tables**

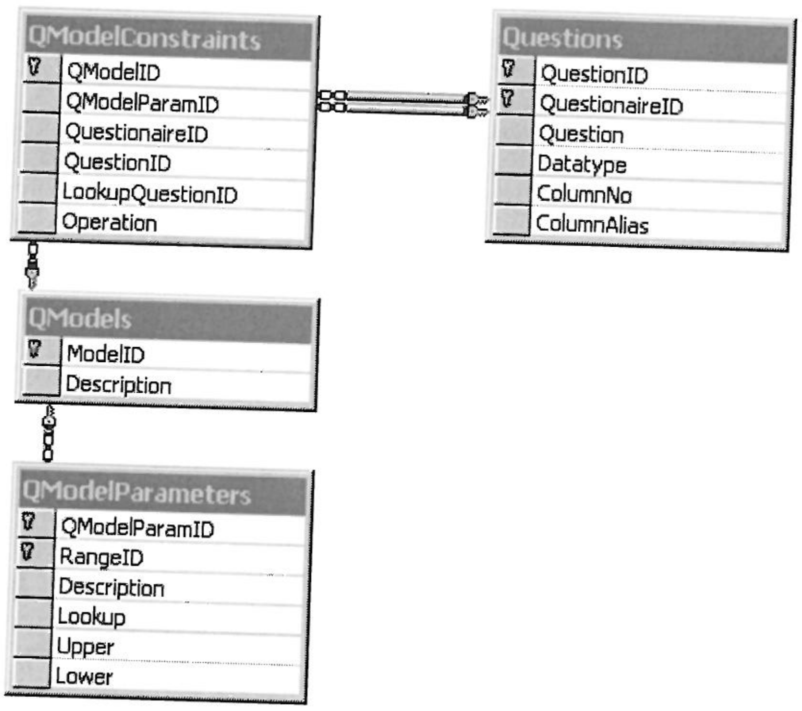


Figure 20 Relationships between tables for storing model type rules

A foreign key relationship exist between *Questions* and *QRedundancy* on QuestionID and QuestionaireID.

9 LOAD DATA QUALITY RULES

A number of tables are used in the specification of the load data quality rules. These can be grouped as either

- Rule definition
- Test result
- Other

For more information see the user guide of LQMarker.EXE

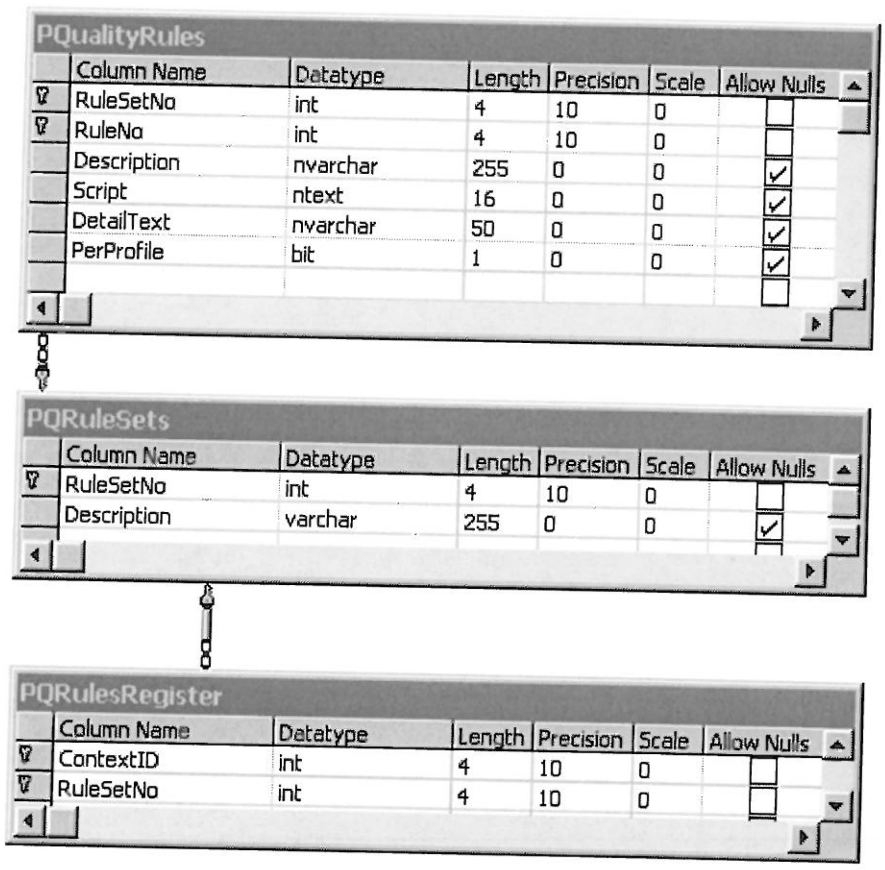
9.1 *Rule definition*

The rule definitions are stored in the PqualityRules table. The script contains a standard SQL92 *script* which are run either per group or per profile depending on the value of the *perprofile* field. A *description* and *detailtext* can be provided to make the results of the rules clearer.

A list of valid *rulesetno's* are stored in PQRuleset together with a description of the ruleset

A ruleset is a set of rules that are used at the same time and can be associated with a context through the PQRulesRegiseter.

The structure of the tables are detailed below.



PQualityRules

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	RuleSetNo	int	4	10	0	☐
🔑	RuleNo	int	4	10	0	☐
	Description	nvarchar	255	0	0	✓
	Script	ntext	16	0	0	✓
	DetailText	nvarchar	50	0	0	✓
	PerProfile	bit	1	0	0	✓

PQRuleSets

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	RuleSetNo	int	4	10	0	☐
	Description	varchar	255	0	0	✓

PQRulesRegister

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	ContextID	int	4	10	0	☐
🔑	RuleSetNo	int	4	10	0	☐

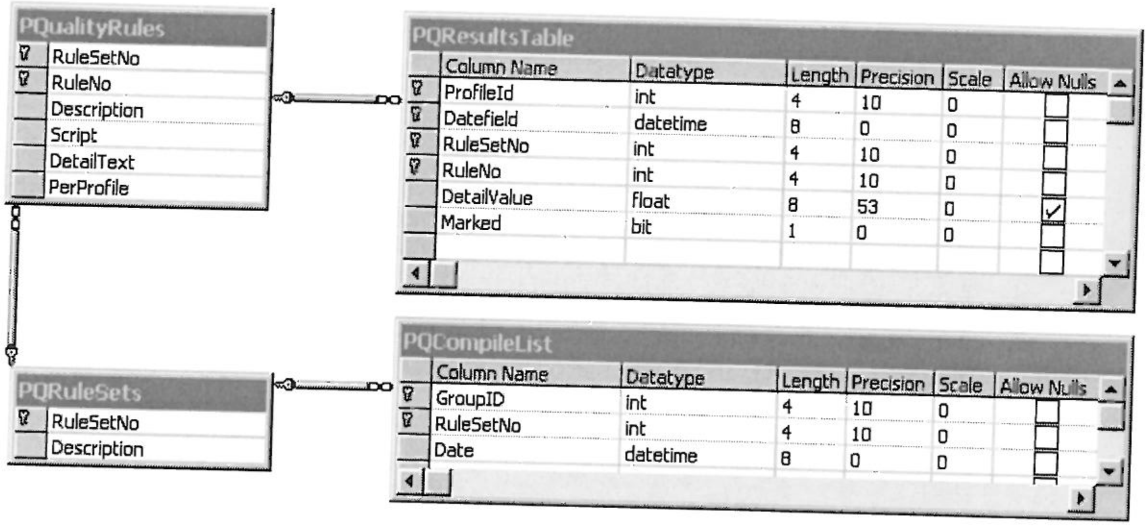
Figure 21 Attributes of the PQualityRules, PQRulesets and PQRulesRegister tables

A Foreign key relationship exists between PqualityRules and PQRuleSets on *RuleSetno.*
A Foreign key relationship exists between PQRulesRegister and PQRuleSets on *RuleSetNo.*
A Foreign key relationship exists between PQRulesRegister and Context on *ContextID.*

9.2 *Test result*

The results of a load quality test on a specific profile are stored in the PQResultsTable. A *datefield profileid* pair stores dates (measured per day) for a specific profile when a specific rule failed. A rule is specified as a *RuleNo* and *RuleSetNo.*

The PQCompileList is used to keep track of when a ruleset was last compiled for a specific group.



PQualityRules

🔑	RuleSetNo
🔑	RuleNo
	Description
	Script
	DetailText
	PerProfile

PQResultsTable

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	ProfileId	int	4	10	0	☐
🔑	Datefield	datetime	8	0	0	☐
🔑	RuleSetNo	int	4	10	0	☐
🔑	RuleNo	int	4	10	0	☐
	DetailValue	float	8	53	0	✓
	Marked	bit	1	0	0	☐

PQRuleSets

🔑	RuleSetNo
	Description

PQCompileList

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	GroupID	int	4	10	0	☐
🔑	RuleSetNo	int	4	10	0	☐
	Date	datetime	8	0	0	☐

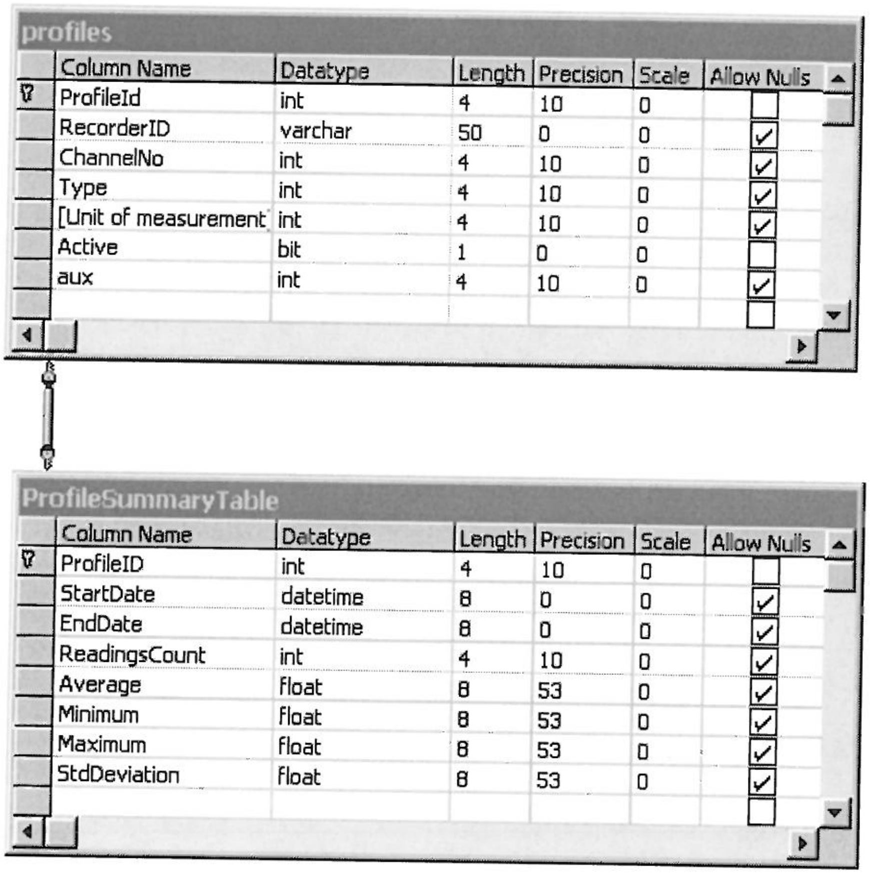
Figure 22 The attributes of the PQResults and the PQCompileList tables

A foreign key relationship exists between the PQResultsTable and the PqualityRules table on *RuleSetno* and *RuleNo.*
A foreign key relationship exists between the PQRuleSets table and the PQCompileList table on *RuleSetno.*
A foreign key relationship exits between the PQCompileList table and Groups table on *groupid.*

9.3 Other

During the compile process of the quality rules, profile summary information is compiled into the ProfileSummaryTable. The following summary information is compiled per *profileid:*

- Start date
- End date
- Reading Count
- Average
- Minimum
- Maximum
- Standard deviation



profiles

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	ProfileId	int	4	10	0	☐
	RecorderID	varchar	50	0	0	✓
	ChannelNo	int	4	10	0	✓
	Type	int	4	10	0	✓
	[Unit of measurement	int	4	10	0	✓
	Active	bit	1	0	0	☐
	aux	int	4	10	0	✓
						☐

ProfileSummaryTable

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls
🔑	ProfileID	int	4	10	0	☐
	StartDate	datetime	8	0	0	✓
	EndDate	datetime	8	0	0	✓
	ReadingsCount	int	4	10	0	✓
	Average	float	8	53	0	✓
	Minimum	float	8	53	0	✓
	Maximum	float	8	53	0	✓
	StdDeviation	float	8	53	0	✓
						☐

Figure 23 Attributes of profilesummarytable

A foreign key relationship exists between Profiles and ProfileSummaryTable on *profileid*.

10 AUXILIARY TABLES

Two auxiliary tables are used by the following software modules:

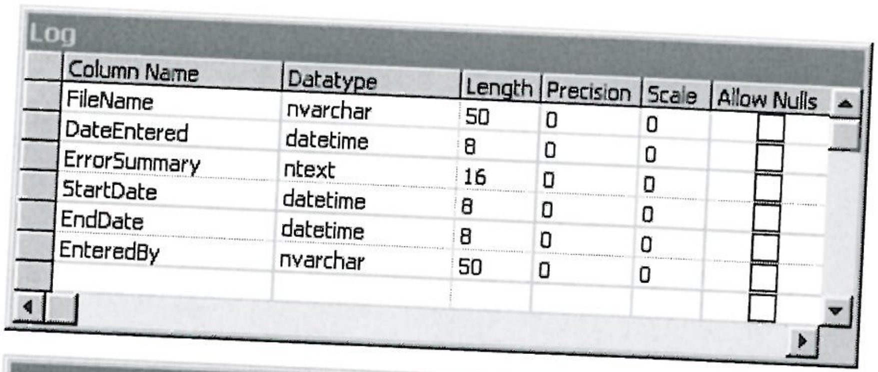
- Log is used by ImpProf.EXE
- NavigatorSetup is used by LRNavigator.EXE

For more information see the relevant user guides.

Log is a simple log of imported files where the following are stored:

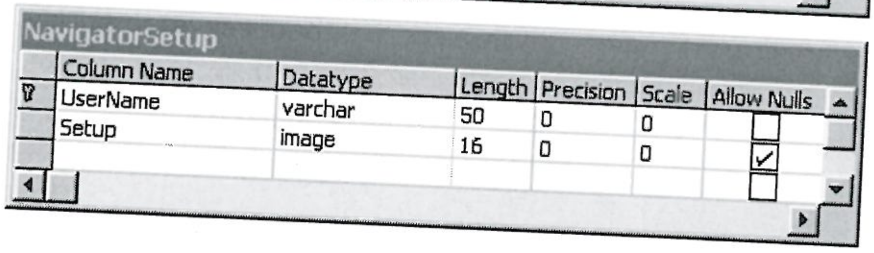
- Filename
- The date the data was loaded
- A list of errors and counts
- First date with data in the loaded file
- Last date with data in the loaded file
- The login of the user that loaded that file

Navigatorsetup contains a binary field where *setup* information per *username* is stored.



Log

Column Name	Datatype	Length	Precision	Scale	Allow Nulls
FileName	nvarchar	50	0	0	☐
DateEntered	datetime	8	0	0	☐
ErrorSummary	ntext	16	0	0	☐
StartDate	datetime	8	0	0	☐
EndDate	datetime	8	0	0	☐
EnteredBy	nvarchar	50	0	0	☐



NavigatorSetup

Column Name	Datatype	Length	Precision	Scale	Allow Nulls
UserName	varchar	50	0	0	☐
Setup	image	16	0	0	✓

Figure 24 Attributes of the Log and NavigatorSetup tables