APPENDIX B

THE STANDARD QUERY SET

TABLE OF CONTENTS

1 INTRODUCTION

This Appendix documents all of the permanent queries which are used on the server side of the data base.

These permanent queries may take 3 forms:

- Stored procedures
- Views
- Triggers on tables

A "stored procedure" is a procedural query which can pass and return variables, much like a subroutine in Visual basic. Stored procedures are often used to standardise procedures on the database so that changes in a client can be kept independent of the underlying data structure on the server.

A "View" is a query which only produces a result set. A view cannot be used to change underlying data.

A "Trigger" is a stored procedure that is initiated when some action takes place on a particular table of the database (i.e. insert or update, or delete). Typically such queries may be used for auditing/logging, security or housekeeping functions on the server data.

2 STORED PROCEDURES

Serial	Stored procedure Name	Used by
1	DeleteAnswerID	Linker
2	DeleteLink	Linker
3	DeleteProfileID	Linker
4	DeleteRuleSet	LQ Marker
5	InsertAnswerID	Socio Capture form
6	InsertLink	ClientDM
7	InsertProfileId	Profile importer
8	[Invalidate entire profile]	LQ Marker
9	[Mark Group based on PQ compile]	LQ Marker
10	[Mark Profile]	LQ Marker
11	[Validate entire profile]	LQ Marker

2.1 Stored procedure DeleteAnswerID

Function: Deletes all trace of a consumer questionnaire response from the database.

```
Alter Procedure DeleteAnswerID
 (
  @AnswerID int
 )
As
begin tran
delete from Answers_char where AnswerID=@AnswerID
if @@error!=0
    begin
    RollBack tran
    return
    end
delete from Answers_blob where AnswerID=@AnswerID
if @@error!=0
    begin
    RollBack tran
    return
    end
delete from Answers_number where AnswerID=@AnswerID
if @@error!=0
    begin
    RollBack tran
    return
    end
delete from LinkTable where AnswerID=@AnswerID
if @@error!=0
    begin
    RollBack tran
    return
    end
Delete from Answers where AnswerID=@AnswerID
if @@error!=0
    begin
    RollBack tran
    return
    end
Commit Tran
```

2.2 Stored procedure DeleteLink

Function: Breaks a specified consumer response to profile link in the Linktable.

```
Alter Procedure DeleteLink
 (
  @AnswerID int,
  @ProfileID int,
  @GroupID int
 )
As
if @ProfileId=0 And @AnswerID=0
    begin
    return
    end
if @GroupId=0
    begin
    return
    end
begin tran
delete from LinkTable where ProfileID=@ProfileID and AnswerID=@AnswerID
if @@error!=0
    begin
    RollBack tran
    return
    end
Commit tran
return
```

2.3 Stored procedure DeleteProfileID

Function: Deletes a profile, with all of its data. ?? what about "profile table"

```
Alter Procedure DeleteProfileID
 (
  @ProfileID int
 )
As
begin tran
delete from LinkTable where ProfileID=@ProfileID
if @@error!=0
    begin
    RollBack tran
    return
    end
Delete from Profiles where ProfileID=@ProfileID
if @@error!=0
    begin
    RollBack tran
    return
    end
Commit Tran
```

2.4 Stored procedure DeleteRuleSet

Function: Delete an entire rule-set and compiled result from a database.

```
Alter Procedure DeleteRuleSet
( @RuleSetNo int )
As
begin tran
Delete from PQRulesRegister where RuleSetNo = @RuleSetNo
if @@error!=0
    begin
    RollBack tran
    return
    end
Delete from PQResultsTable where RuleSetNo = @RuleSetNo
if @@error!=0
    begin
    RollBack tran
    return
    end
Delete from PQualityRules where RuleSetNo = @RuleSetNo
if @@error!=0
    begin
    RollBack tran
    return
    end
Delete from PQCompileList where RuleSetNo = @RuleSetNo
if @@error!=0
    begin
    RollBack tran
    return
    end
Delete from PQRuleSets where RuleSetNo = @RuleSetNo
if @@error!=0
    begin
    RollBack tran
    return
    end
        Commit Tran
```

2.5 Stored procedure InsertAnswerID

Function: Creates a new consumer response.

```
Alter Procedure InsertAnswerID
(
 @AnswerID int,
 @QuestionaireID int,
 @GroupID int
)
As
begin tran
insert into Answers (AnswerID,QuestionaireID) values
(@AnswerID,@QuestionaireID)
if @@error!=0
   begin
   Rollback Tran
   print 'Insert failed'
   return
   end
insert into Answers_blob (AnswerID) values (@AnswerID)
if @@error!=0
   begin
   Rollback Tran
   print 'Insert failed'
   return
   end
insert into Answers_char (AnswerID) values (@AnswerID)
if @@error!=0
   begin
   Rollback Tran
   print 'Insert failed'
   return
   end
insert into Answers_number (AnswerID) values (@AnswerID)
if @@error!=0
   begin
   Rollback Tran
   print 'Insert failed'
   return
   end
insert into Linktable (AnswerID,ConsumerID,ProfileID,GroupID) values
(@AnswerID,0,0,@GroupID)
if @@error!=0
   begin
   Rollback Tran
   print 'Insert failed'
```

```
    return
    end
Commit Tran
```

2.6 Stored procedure InsertLink

Function: Creates a link between a Profile and a consumer response.

```
Alter Procedure InsertLink
(
 @AnswerID int,
 @ProfileID int,
 @GroupID int
)
As
if @ProfileId=0 And @AnswerID=0
    begin
    return
    end
if @GroupId=0
    begin
    return
    end
begin tran
insert into LinkTable (ConsumerID,AnswerID,ProfileID,GroupID) Values
(0,@AnswerID,@ProfileID,0)
if @@error!=0
    begin
    RollBack tran
    return
    end
Commit tran
/* set nocount on */
return
```

2.7 Stored procedure InsertProfileId

Function: Creates a new load profile.

```
Alter Procedure InsertProfileId
(
 @ProfileID int,
 @GroupID int
)
As
begin tran
insert into Profiles (ProfileID) values (@ProfileID)
if @@error!=0
   begin
   Rollback Tran
   print 'Insert failed'
   return
   end
insert into Linktable (AnswerID,ConsumerID,ProfileID,GroupID) values
(0,0,@ProfileID,@GroupID)
if @@error!=0
   begin
   Rollback Tran
   print 'Insert failed'
   return
   end
Commit Tran
```

2.8 Stored procedure [Invalidate entire profile]

Function: Marks an entire profile invalid. (1) each individual reading in the ProfileTable is marked 'bad' , and (2) The entire profile is flagged inactive and (3) The PQResults table is updated with the result.

```
Alter Procedure [Invalidate entire profile]
        (@ProfileID int, @RuleSetNo int      )
As
begin tran
update ProfileTable set Valid = 'N'  where profileid = @ProfileID
if @@error<>0
   begin
   rollback tran
   return
   end

update profiles set active = 0  where profileid = @ProfileID
if @@error<>0
   begin
   rollback tran
   return
   end

update PQResultsTable set Marked = -1 where profileid = @ProfileID and
RuleSetNo = @RuleSetNo
if @@error<>0
   begin
   rollback tran
   return
   end

commit tran
```

2.9 Stored procedure [Mark Group based on PQ compile]

Function: The results of the latest PQ compile are use to flag the data points in the profile table as good & bad. Data marking is carried out on a day-block basis. This is an automatic mode of data quality marking.

```
Alter Procedure [Mark Group based on PQ compile]
( @groupid int, @RuleSetNo int )
As
begin tran

update ProfileTable set Valid= 'Y'
 From ProfileTable inner join Linktable on
ProfileTable.ProfileId=LinkTable.ProfileID
inner join profiles on profiletable.profileid = profiles.profileid
 where (LinktAble.GroupId=@GroupID )
and (profiles.Active = 1)

if @@error <>0
begin
 rollback tran
 return
end

update ProfileTable set Valid= 'N'
 From ProfileTable inner join Linktable on
ProfileTable.ProfileId=LinkTable.ProfileID
inner join profiles on profiletable.profileid = profiles.profileid
 where (LinktAble.GroupId=@GroupID )
and (profiles.Active =0)

if @@error <>0
begin
 rollback tran
 return
end

update ProfileTable set Valid= 'N'
 From ProfileTable inner join Linktable on
ProfileTable.ProfileId=LinkTable.ProfileID
 inner join PQResultsTable on
 ProfileTable.DateField between PQResultsTable.Datefield and
DateAdd(mi,60*24-1,PQResultsTable.DateField)
 and
ProfileTable.ProfileID=PQResultsTable.ProfileID
 where (LinktAble.GroupId=@GroupID )
```

```
and (PQResultsTable.RuleSetNo=@RuleSetNo)
if @@error <>0
begin
 rollback tran
 return
end

Update PQResultsTable set Marked = -1 from
PQResultsTable inner join linktable on
PQResultsTable.ProfileID=LinkTable.ProfileID
where LinkTable.GroupID = @GroupID and RuleSetNo= @RuleSetNo
if @@error <>0
begin
 rollback tran
 returnend
commit tran
```

2.10 Stored procedure [Mark Profile]

Function: Readings in a profile are flagged valid/invalid. Data marking is carried out on a day-block basis. Passing of rule-set no.??

```
Alter Procedure [Mark Profile]
( @profileid int, @year int, @month int, @day int, @Active bit, @RuleSetNo int )
As
begin tran

if @Active = -1
 begin
update ProfileTable set Valid= 'Y' From ProfileTable where ProfileID=@ProfileID
and
 DateAdd(dd,DateDiff(dd,0,DateField),0) = DateAdd(yy,@Year-1900,
DateAdd(mm,@Month-1,DateAdd(dd,@Day-1,0)))
end
else
begin
update ProfileTable set Valid= 'N' From ProfileTable where ProfileID=@ProfileID
and
 DateAdd(dd,DateDiff(dd,0,DateField),0) = DateAdd(yy,@Year-
1900,DateAdd(mm,@Month-1,DateAdd(dd,@Day-1,0)))
end

if @@error <>0
begin
 rollback tran
 return
end

if @Active = -1
 begin
 Update PQResultsTable set Marked = 0 where ProfileID = @ProfileID
 and DateField = DateAdd(yy,@Year-1900,DateAdd(mm,@Month-
1,DateAdd(dd,@Day-1,0)))
 end
else
 begin
 Update PQResultsTable set Marked = -1 where ProfileID = @ProfileID
 and Datefield = DateAdd(yy,@Year-1900,DateAdd(mm,@Month-
1,DateAdd(dd,@Day-1,0)))
 end

if @@error <>0
begin
 rollback tran
```

```
  return
end

commit tran
```

2.11 Stored procedure [Validate entire profile]

Function: Turns an entire profile to "valid" setting.

```
Alter Procedure [Validate entire profile]
        (@ProfileID int,@RuleSetNo int      )
As
begin tran
update ProfileTable set Valid = 'Y' where profileid = @ProfileID
if @@error<>0
   begin
   rollback tran
   return
   end

update profiles set active = -1  where profileid = @ProfileID
if @@error<>0
   begin
   rollback tran
   return
   end

update PQResultsTable set Marked = 0 where profileid = @ProfileID and
rulesetno=@RuleSetNo
if @@error<>0
   begin
   rollback tran
   return
   end
commit tran
```

3 SUMMARY OF VIEWS

Serial	View name	Used by
1	Constraintsview	Socio quality checker
2	RedundancyView	Socio quality checker
3		
4		
5		
6		
7		
8		
9		
10		
11		

3.1 Constraints view

```
CREATE VIEW dbo.ConstraintsView
AS
SELECT QConstraints.QCID, QConstraints.Lower,
   QConstraints.Upper, QConstraints.AllowNull,
   Questions.ColumnAlias, QConstraints.LookupQuestionID,
   Questions1.ColumnAlias AS LookupFieldname,
   QConstraints.QuestionaireID
FROM Questions INNER JOIN
   QConstraints ON
   Questions.QuestionID = QConstraints.QuestionID AND
   Questions.QuestionaireID = QConstraints.QuestionaireID INNER
    JOIN
   Questions Questions1 ON
   QConstraints.QuestionaireID = Questions1.QuestionaireID AND
   QConstraints.LookupQuestionID = Questions1.QuestionID
```

3.2 Redundancy view

```
CREATE VIEW dbo.RedundancyView
AS
SELECT QRedundancy.QRID, QRedundancy.Operation,
   Questions.ColumnAlias,
   Questions1.ColumnAlias AS LookupFieldName,
   QRedundancy.QuestionaireId, QRedundancy.UpperTolerance,
   QRedundancy.LowerTolerance,
   QRedundancy.UseTolerance
FROM QRedundancy INNER JOIN
   Questions ON
   QRedundancy.QuestionID = Questions.QuestionID AND
   QRedundancy.QuestionaireId = Questions.QuestionaireID INNER
   JOIN
   Questions Questions1 ON
   QRedundancy.QuestionaireId = Questions1.QuestionaireID AND
   QRedundancy.LookupQuestionID = Questions1.QuestionID
```

4 SUMMARY OF TRIGGERS

Serial	Trigger Name	Initiated by:
1	linktable_Trigger1	Linktable

4.1 Trigger linktable_Trigger1

Function: Deletes any "null" links left on the linktable.

```
Alter Trigger linktable_Trigger1
On dbo.linktable
For  Insert, Update
As
 Delete from Linktable where Profileid=0 and AnswerID=0
```

4.2 _<TableName>_DeleteLock

There is one '_DeleteLock' trigger for each table in the database that forms part of the locking implementation. Refer to the DBImportExport manual for a detailed description of these triggers.

4.3 _<TableName>_InsertLock

There is one '_InsertLock' trigger for each table in the database that forms part of the locking implementation. Refer to the DBImportExport manual for a detailed description of these triggers.