```
unit DataElements;

interface

uses dialogs, Generics.Collections, LCRGlobals,
     DB, ADODB,
     FireDAC.Comp.Client, FireDAC.Phys.MSAcc, FireDAC.Stan.Def, FireDAC.DApt,
     FireDAC.Stan.Async, FireDAC.Stan.Param;

type

TDataElement = class
public
  name: string;
  data_type: TDataNeedType;
  cost_base: TCostBase;
  unit_basis: TUnitBasis;
  strata_system_size: boolean;
  strata_source_water: boolean;
  strata_lsl: boolean;
  strata_cct: boolean;
  distribution: TDistributionType;

  StrataDistribution: string;
  HasData: boolean;
  StrataType: string;
  BaselineSame: boolean;

  //FAccessDB: TFDConnection;
  //FAccessQry: TFDQuery;
  FAccessDB: TADOConnection;
  FAccessQry: TADOQuery;

  constructor Create; overload;
  constructor Create(AName, ADataType, ACostBase, AUnitBasis, AStrataSystemSize,
                     AStrataSourceWater, AStrataLsl, AStrataCct, ADistribution:
string); overload;
  destructor Destroy; override;

  procedure ReadData;
  function GetValue(vType: string; iSystemSize, iSourceWater, iLSL, iCCT: integer):
double;
private
  queryType: string;
  querySQL: string;
  arrayType: string;

  vXValue: double;
  vMinValue: double;
  vMaxValue: double;
```

```
vMostLikely: double;
vMean: double;
vStdDev: double;
vFunction: string;
vCustom: double;

YNNNPointEstimate: TDoubleArray;
YNNNMinimum: TDoubleArray;
YNNNMaximum: TDoubleArray;
YNNNMostLikely: TDoubleArray;
YNNNMean: TDoubleArray;
YNNNStdDev: TDoubleArray;

YYNNPointEstimate: TDoubleArray2;

YYYNPointEstimate: TDoubleArray3;
YYYNMinimum: TDoubleArray3;
YYYNMaximum: TDoubleArray3;
YYYNMostLikely: TDoubleArray3;
YYYNMean: TDoubleArray3;
YYYNStdDev: TDoubleArray3;

YYNYPointEstimate: TDoubleArray3;
YYNYMinimum: TDoubleArray3;
YYNYMaximum: TDoubleArray3;
YYNYMostLikely: TDoubleArray3;
YYNYMean: TDoubleArray3;
YYNYStdDev: TDoubleArray3;

YNYYPointEstimate: TDoubleArray3;
YNYYMinimum: TDoubleArray3;
YNYYMaximum: TDoubleArray3;
YNYYMostLikely: TDoubleArray3;
YNYYMean: TDoubleArray3;
YNYYStdDev: TDoubleArray3;

YNYNPointEstimate: TDoubleArray2;
YNYNMinimum: TDoubleArray2;
YNYNMaximum: TDoubleArray2;
YNYNMostLikely: TDoubleArray2;
YNYNMean: TDoubleArray2;
YNYNStdDev: TDoubleArray2;

YNNYPointEstimate: TDoubleArray2;
YNNYMinimum: TDoubleArray2;
YNNYMaximum: TDoubleArray2;
YNNYMostLikely: TDoubleArray2;
YNNYMean: TDoubleArray2;
YNNYStdDev: TDoubleArray2;
```

```
  YYYYPointEstimate: TDoubleArray4;
  YYYYMinimum: TDoubleArray4;
  YYYYMaximum: TDoubleArray4;
  YYYYMostLikely: TDoubleArray4;
  YYYYMean: TDoubleArray4;
  YYYYStdDev: TDoubleArray4;

  NYYNPointEstimate: TDoubleArray2;
  NYYNMinimum: TDoubleArray2;
  NYYNMaximum: TDoubleArray2;
  NYYNMostLikely: TDoubleArray2;
  NYYNMean: TDoubleArray2;
  NYYNStdDev: TDoubleArray2;

  NNYNPointEstimate: TDoubleArray;
  NNYNMinimum: TDoubleArray;
  NNYNMaximum: TDoubleArray;
  NNYNMostLikely: TDoubleArray;
  NNYNMean: TDoubleArray;
  NNYNStdDev: TDoubleArray;

  NYNNPointEstimate: TDoubleArray;
  NYNNMinimum: TDoubleArray;
  NYNNMaximum: TDoubleArray;
  NYNNMostLikely: TDoubleArray;
  NYNNMean: TDoubleArray;
  NYNNStdDev: TDoubleArray;

  NYNYPointEstimate: TDoubleArray2;
  NYNYMinimum: TDoubleArray2;
  NYNYMaximum: TDoubleArray2;
  NYNYMostLikely: TDoubleArray2;
  NYNYMean: TDoubleArray2;
  NYNYStdDev: TDoubleArray2;

  NNNYPointEstimate: TDoubleArray;
  NNNYMinimum: TDoubleArray;
  NNNYMaximum: TDoubleArray;
  NNNYMostLikely: TDoubleArray;
  NNNYMean: TDoubleArray;
  NNNYStdDev: TDoubleArray;

  procedure SetQueryType;
  function GetQuerySQL: string;
  procedure ExecQuery;
end;

TDataElements = class
public
  DataStore: TObjectDictionary<string, TDataElement>;
```

```
  FAccessDB: TADOConnection;
  FAccessQry: TADOQuery;

  procedure Add(key: string; de: TDataElement);
  function GetVariable(varname: string): TDataElement;
  function BaselineSame(varname: string): boolean;

  constructor Create(ADataPath: string);
  destructor Destroy; override;

private
  DataPath: string;
end;

implementation

uses SysUtils, StrUtils;

{ TDataElement }

constructor TDataElement.Create;
begin

end;

constructor TDataElement.Create(AName, ADataType, ACostBase, AUnitBasis,
AStrataSystemSize,
  AStrataSourceWater, AStrataLsl, AStrataCct, ADistribution: string);
begin
  name := AName;
  data_type := TDataNeedTypeFromStr(ADataType);
  cost_base := TCostBaseFromStr(ACostBase);
  unit_basis := TUnitBasisFromStr(AUnitBasis);
  if AStrataSystemSize = 'Y' then strata_system_size := True else strata_system_size
:= False;
  if AStrataSourceWater = 'Y' then strata_source_water := True else
strata_source_water := False;
  if AStrataLsl= 'Y' then strata_lsl := True else strata_lsl := False;
  if AStrataCct = 'Y' then strata_cct := True else strata_cct := False;
  distribution := TDistributionTypeFromStr(ADistribution);

  SetQueryType;

  {
    QueryType:
    NNNNPoint Estimate
    NNNNTriangular
    NNNNUniform
    YNNNPoint Estimate
```

```
  YNNNTriangular
  YNNNUniform
  YYNNPoint Estimate
  YYNNFunction
  YYYNPoint Estimate
  YYYNTriangular
  YYYNUniform
  YYNYPoint Estimate
  YNYYPoint Estimate
  YNYYUniform
  YNYYTriangular
  YNYNPoint Estimate
  YNYNTriangular
  YNYNUniform
  YNNYPoint Estimate
  YNNYUniform
  YYYYPoint Estimate
  YYYYTriangular
  YYYYUniform
  NYYNPoint Estimate
  NYYNTriangular
  NNYNPoint Estimate
  NNYNTriangular
  NYNNPoint Estimate
  NYNNUniform
  NYNYPoint Estimate
  NNNYPoint Estimate

  ArrayType:
  NNNN
  YNYY
  YNYN
  YNNY
  YNNN
  YYYN
  YYNY
  YYYY
  NYYN
  NNYN
  NYNN
  NYNY
  NNNY
}
StrataDistribution := queryType;

if QueryType = 'NNNNPoint Estimate' then
begin
end
else if QueryType = 'NNNNTriangular' then
begin
```

```
end
else if QueryType = 'NNNNUniform' then
begin
end
else if QueryType = 'YYNNPoint Estimate' then
begin
  SetLength(YYNNPointEstimate, 9, 3);
end
else if QueryType = 'YYNYPoint Estimate' then
begin
  SetLength(YYNYPointEstimate, 9, 3, 2);
end
else if QueryType = 'YYNNFunction' then
begin
end
else if QueryType = 'YNNNPoint Estimate' then
begin
  SetLength(YNNNPointEstimate, 9);
end
else if QueryType = 'YNNNTriangular' then
begin
  SetLength(YNNNMinimum, 9);
  SetLength(YNNNMaximum, 9);
  SetLength(YNNNMostLikely, 9);
end
else if QueryType = 'YNNNUniform' then
begin
  SetLength(YNNNMinimum, 9);
  SetLength(YNNNMaximum, 9);
end
else if QueryType = 'YYYNPoint Estimate' then
begin
  SetLength(YYYNPointEstimate, 9, 3, 2);
end
else if QueryType = 'YYYNTriangular' then
begin
  SetLength(YYYNMinimum, 9, 3, 2);
  SetLength(YYYNMaximum, 9, 3, 2);
  SetLength(YYYNMostLikely, 9, 3, 2);
end
else if QueryType = 'YYYNUniform' then
begin
  SetLength(YYYNMinimum, 9, 3, 2);
  SetLength(YYYNMaximum, 9, 3, 2);
end
else if QueryType = 'YNYYPoint Estimate' then
begin
  SetLength(YNYYPointEstimate, 9, 2, 2);
end
else if QueryType = 'YNYYUniform' then
```

```
begin
  SetLength(YNYYMinimum, 9, 2, 2);
  SetLength(YNYYMaximum, 9, 2, 2);
end
else if QueryType = 'YNYYTriangular' then
begin
  SetLength(YNYYMinimum, 9, 2, 2);
  SetLength(YNYYMaximum, 9, 2, 2);
  SetLength(YNYYMostLikely, 9, 2, 2);
end
else if QueryType = 'YNYNPoint Estimate' then
begin
  SetLength(YNYNPointEstimate, 9, 2);
end
else if QueryType = 'YNYNTriangular' then
begin
  SetLength(YNYNMinimum, 9, 2);
  SetLength(YNYNMaximum, 9, 2);
  SetLength(YNYNMostLikely, 9, 2);
end
else if QueryType = 'YNYNUniform' then
begin
  SetLength(YNYNMinimum, 9, 2);
  SetLength(YNYNMaximum, 9, 2);
end
else if QueryType = 'YNNYPoint Estimate' then
begin
  SetLength(YNNYPointEstimate, 9, 2);
end
else if QueryType = 'YNNYUniform' then
begin
  SetLength(YNNYMinimum, 9, 2);
  SetLength(YNNYMaximum, 9, 2);
end
else if QueryType = 'YYYYPoint Estimate' then
begin
  SetLength(YYYYPointEstimate, 9, 3, 2, 2);
end
else if QueryType = 'YYYYTriangular' then
begin
  SetLength(YYYYMinimum, 9, 3, 2, 2);
  SetLength(YYYYMaximum, 9, 3, 2, 2);
  SetLength(YYYYMostLikely, 9, 3, 2, 2);
end
else if QueryType = 'YYYYUniform' then
begin
  SetLength(YYYYMinimum, 9, 3, 2, 2);
  SetLength(YYYYMaximum, 9, 3, 2, 2);
end
else if QueryType = 'NYYNPointEstimate' then
```

```
  begin
    SetLength(NYYNPointEstimate, 3, 2);
  end
  else if QueryType = 'NYYNTriangular' then
  begin
    SetLength(NYYNMinimum, 3, 2);
    SetLength(NYYNMaximum, 3, 2);
    SetLength(NYYNMostLikely, 3, 2);
  end
  else if QueryType = 'NNYNPoint Estimate' then
  begin
    SetLength(NNYNPointEstimate, 2);
  end
  else if QueryType = 'NYNNPoint Estimate' then
  begin
    SetLength(NYNNPointEstimate, 3);
  end
  else if QueryType = 'NYNNUniform' then
  begin
    SetLength(NYNNMinimum, 3);
    SetLength(NYNNMaximum, 3);
  end
  else if QueryType = 'NNYNTriangular' then
  begin
    SetLength(NNYNMinimum, 2);
    SetLength(NNYNMaximum, 2);
    SetLength(NNYNMostLikely, 2);
  end
  else if QueryType = 'NYNYPoint Estimate' then
  begin
    SetLength(NYNYPointEstimate, 3, 2);
  end
  else if QueryType = 'NNNYPoint Estimate' then
  begin
    SetLength(NNNYPointEstimate, 2);
  end
  else
  raise Exception.Create('Invalid QueryType value: ' + QueryType);

  querySQL := GetQuerySQL;

  HasData := false;
end;

destructor TDataElement.Destroy;
begin
  //ShowMessage(name + ' destroyed');
  inherited;
end;
```

```
procedure TDataElement.ExecQuery;
begin
  FAccessQry.SQL.Clear;
  FAccessQry.SQL.Add(querySql);
  FAccessQry.Parameters.ParamByName('ID').Value := name;
  FAccessQry.Open;
  if not FAccessQry.Eof then
  begin
    HasData := true;
    while not FAccessQry.Eof do
    begin

      if QueryType = 'NNNNPoint Estimate' then
      begin
        vXValue := FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'NNNNTriangular' then
      begin
        vMinValue := FAccessQry.FieldByName('MinValue').AsFloat;
        vMaxValue := FAccessQry.FieldByName('MaxValue').AsFloat;
        vMostLikely := FAccessQry.FieldByName('MostLikely').AsFloat;
      end
      else if QueryType = 'NNNNUniform' then
      begin
        vMinValue := FAccessQry.FieldByName('MinValue').AsFloat;
        vMaxValue := FAccessQry.FieldByName('MaxValue').AsFloat;
      end
      else if QueryType = 'YYNNPoint Estimate' then
      begin
        YYNNPointEstimate[FAccessQry.FieldByName('SystemSize').AsInteger,
                          FAccessQry.FieldByName('SourceWater').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'YYNNFunction' then
      begin
        vFunction := FAccessQry.FieldByName('Function').AsString;
      end
      else if QueryType = 'YNNNPoint Estimate' then
      begin
        YNNNPointEstimate[FAccessQry.FieldByName('SystemSize').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'YNNNTriangular' then
      begin
        YNNNMinimum[FAccessQry.FieldByName('SystemSize').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        YNNNMaximum[FAccessQry.FieldByName('SystemSize').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
        YNNNMostLikely[FAccessQry.FieldByName('SystemSize').AsInteger] :=
FAccessQry.FieldByName('MostLikely').AsFloat;
```

```
      end
      else if QueryType = 'YNNNUniform' then
      begin
        YNNNMinimum[FAccessQry.FieldByName('SystemSize').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        YNNNMaximum[FAccessQry.FieldByName('SystemSize').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
      end
      else if QueryType = 'YYYNPoint Estimate' then
      begin
        YYYNPointEstimate[FAccessQry.FieldByName('SystemSize').AsInteger,
                          FAccessQry.FieldByName('SourceWater').AsInteger,
                          FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'YYYNTriangular' then
      begin
        YYYNMinimum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('SourceWater').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        YYYNMaximum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('SourceWater').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
        YYYNMostLikely[FAccessQry.FieldByName('SystemSize').AsInteger,
                       FAccessQry.FieldByName('SourceWater').AsInteger,
                       FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MostLikely').AsFloat;
      end
      else if QueryType = 'YYYNUniform' then
      begin
        YYYNMinimum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('SourceWater').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        YYYNMaximum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('SourceWater').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
      end
      else if QueryType = 'YYNYPoint Estimate' then
      begin
        YYNYPointEstimate[FAccessQry.FieldByName('SystemSize').AsInteger,
                          FAccessQry.FieldByName('SourceWater').AsInteger,
                          FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'YNYYPoint Estimate' then
      begin
```

```
        YNYYPointEstimate[FAccessQry.FieldByName('SystemSize').AsInteger,
                          FAccessQry.FieldByName('LSL').AsInteger,
                          FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'YNYYUniform' then
      begin
        YNYYMinimum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger,
                    FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        YNYYMaximum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger,
                    FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
      end
      else if QueryType = 'YNYYTriangular' then
      begin
        YNYYMinimum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger,
                    FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        YNYYMaximum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger,
                    FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
        YNYYMostLikely[FAccessQry.FieldByName('SystemSize').AsInteger,
                       FAccessQry.FieldByName('LSL').AsInteger,
                       FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MostLikely').AsFloat;
      end
      else if QueryType = 'YNYNPoint Estimate' then
      begin
        YNYNPointEstimate[FAccessQry.FieldByName('SystemSize').AsInteger,
                          FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'YNYNTriangular' then
      begin
        YNYNMinimum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        YNYNMaximum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
        YNYNMostLikely[FAccessQry.FieldByName('SystemSize').AsInteger,
                       FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MostLikely').AsFloat;
      end
      else if QueryType = 'YNYNUniform' then
```

```
      begin
        YNYNMinimum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        YNYNMaximum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
      end
      else if QueryType = 'YNNYPoint Estimate' then
      begin
        YNNYPointEstimate[FAccessQry.FieldByName('SystemSize').AsInteger,
                          FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'YNNYUniform' then
      begin
        YNNYMinimum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        YNNYMaximum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
      end
      else if QueryType = 'YYYYPoint Estimate' then
      begin
        YYYYPointEstimate[FAccessQry.FieldByName('SystemSize').AsInteger,
                          FAccessQry.FieldByName('SourceWater').AsInteger,
                          FAccessQry.FieldByName('LSL').AsInteger,
                          FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'YYYYTriangular' then
      begin
        YYYYMinimum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('SourceWater').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger,
                    FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        YYYYMaximum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('SourceWater').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger,
                    FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
        YYYYMostLikely[FAccessQry.FieldByName('SystemSize').AsInteger,
                       FAccessQry.FieldByName('SourceWater').AsInteger,
                       FAccessQry.FieldByName('LSL').AsInteger,
                       FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MostLikely').AsFloat;
      end
      else if QueryType = 'YYYYUniform' then
```

```
      begin
        YYYYMinimum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('SourceWater').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger,
                    FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        YYYYMaximum[FAccessQry.FieldByName('SystemSize').AsInteger,
                    FAccessQry.FieldByName('SourceWater').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger,
                    FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
      end
      else if QueryType = 'NYYNPointEstimate' then
      begin
        NYYNPointEstimate[FAccessQry.FieldByName('SourceWater').AsInteger,
                          FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'NYYNTriangular' then
      begin
        NYYNMinimum[FAccessQry.FieldByName('SourceWater').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        NYYNMaximum[FAccessQry.FieldByName('SourceWater').AsInteger,
                    FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
        NYYNMostLikely[FAccessQry.FieldByName('SourceWater').AsInteger,
                       FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MostLikely').AsFloat;
      end
      else if QueryType = 'NNYNPoint Estimate' then
      begin
        NNYNPointEstimate[FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'NYNNPoint Estimate' then
      begin
        NYNNPointEstimate[FAccessQry.FieldByName('SourceWater').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'NYNNUniform' then
      begin
        NYNNMinimum[FAccessQry.FieldByName('SourceWater').AsInteger] :=
FAccessQry.FieldByName('MinValue').AsFloat;
        NYNNMaximum[FAccessQry.FieldByName('SourceWater').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
      end
      else if QueryType = 'NNYNTriangular' then
      begin
        NNYNMinimum[FAccessQry.FieldByName('LSL').AsInteger] :=
```

```
FAccessQry.FieldByName('MinValue').AsFloat;
        NNYNMaximum[FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MaxValue').AsFloat;
        NNYNMostLikely[FAccessQry.FieldByName('LSL').AsInteger] :=
FAccessQry.FieldByName('MostLikely').AsFloat;
      end
      else if QueryType = 'NYNYPoint Estimate' then
      begin
        NYNYPointEstimate[FAccessQry.FieldByName('SourceWater').AsInteger,
                          FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else if QueryType = 'NNNYPoint Estimate' then
      begin
        NNYNPointEstimate[FAccessQry.FieldByName('CCT').AsInteger] :=
FAccessQry.FieldByName('XValue').AsFloat;
      end
      else
      raise Exception.Create('Invalid QueryType value: ' + QueryType);

      FAccessQry.Next;
    end;

  end;
  FAccessQry.Close;

end;

function TDataElement.GetQuerySQL;
var
  sSql: string;
begin
  if QueryType = 'NNNNPoint Estimate' then
    sSql := 'SELECT XValue FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'NNNNTriangular' then
    sSql := 'SELECT MinValue, MaxValue, MostLikely FROM InputValues WHERE ID_Name =
:ID'
  else if QueryType = 'NNNNUniform' then
    sSql := 'SELECT MinValue, MaxValue FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'YYNNPoint Estimate' then
    sSql := 'SELECT SystemSize, SourceWater, XValue FROM InputValues WHERE ID_Name =
:ID'
  else if QueryType = 'YYNNFunction' then
    sSql := 'SELECT Function FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'YNNNPoint Estimate' then
    sSql := 'SELECT SystemSize, XValue FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'YNNNTriangular' then
    sSql := 'SELECT SystemSize, MinValue, MaxValue, MostLikely FROM InputValues
WHERE ID_Name = :ID'
  else if QueryType = 'YNNNUniform' then
```

```
    sSql := 'SELECT SystemSize, MinValue, MaxValue FROM InputValues WHERE ID_Name =
:ID'
  else if QueryType = 'YYYNPoint Estimate' then
    sSql := 'SELECT SystemSize, SourceWater, LSL, XValue FROM InputValues WHERE
ID_Name = :ID'
  else if QueryType = 'YYNYPoint Estimate' then
    sSql := 'SELECT SystemSize, SourceWater, CCT, XValue FROM InputValues WHERE
ID_Name = :ID'
  else if QueryType = 'YYYNTriangular' then
    sSql := 'SELECT SystemSize, SourceWater, LSL, MinValue, MaxValue, MostLikely
FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'YYYNUniform' then
    sSql := 'SELECT SystemSize, SourceWater, LSL, MinValue, MaxValue FROM
InputValues WHERE ID_Name = :ID'
  else if QueryType = 'YNYYPoint Estimate' then
    sSql := 'SELECT SystemSize, LSL, CCT, XValue FROM InputValues WHERE ID_Name =
:ID'
  else if QueryType = 'YNYYUniform' then
    sSql := 'SELECT SystemSize, LSL, CCT, MinValue, MaxValue FROM InputValues WHERE
ID_Name = :ID'
  else if QueryType = 'YNYYTriangular' then
    sSql := 'SELECT SystemSize, LSL, CCT, MinValue, MaxValue, MostLikely FROM
InputValues WHERE ID_Name = :ID'
  else if QueryType = 'YNYNPoint Estimate' then
    sSql := 'SELECT SystemSize, LSL, XValue FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'YNYNTriangular' then
    sSql := 'SELECT SystemSize, LSL, MinValue, MaxValue, MostLikely FROM InputValues
WHERE ID_Name = :ID'
  else if QueryType = 'YNYNUniform' then
    sSql := 'SELECT SystemSize, LSL, MinValue, MaxValue FROM InputValues WHERE
ID_Name = :ID'
  else if QueryType = 'YNNYPoint Estimate' then
    sSql := 'SELECT SystemSize, CCT, XValue FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'YNNYUniform' then
    sSql := 'SELECT SystemSize, CCT, MinValue, MaxValue FROM InputValues WHERE
ID_Name = :ID'
  else if QueryType = 'YYYYPoint Estimate' then
    sSql := 'SELECT SystemSize, SourceWater, LSL, CCT, XValue FROM InputValues WHERE
ID_Name = :ID'
  else if QueryType = 'YYYYTriangular' then
    sSql := 'SELECT SystemSize, SourceWater, LSL, CCT, MinValue, MaxValue,
MostLikely FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'YYYYUniform' then
    sSql := 'SELECT SystemSize, SourceWater, LSL, CCT, MinValue, MaxValue FROM
InputValues WHERE ID_Name = :ID'
  else if QueryType = 'NYYNPointEstimate' then
    sSql := 'SELECT SourceWater, LSL, XValue FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'NYYNTriangular' then
    sSql := 'SELECT SourceWater, LSL, MinValue, MaxValue, MostLikely FROM
InputValues WHERE ID_Name = :ID'
```

```
  else if QueryType = 'NNYNPoint Estimate' then
    sSql := 'SELECT LSL, XValue FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'NYNNPoint Estimate' then
    sSql := 'SELECT SourceWater, XValue FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'NYNNUniform' then
    sSql := 'SELECT SourceWater, MinValue, MaxValue FROM InputValues WHERE ID_Name =
:ID'
  else if QueryType = 'NNYNTriangular' then
    sSql := 'SELECT LSL, MinValue, MaxValue, MostLikely FROM InputValues WHERE
ID_Name = :ID'
  else if QueryType = 'NYNYPoint Estimate' then
    sSql := 'SELECT SourceWater, CCT, XValue FROM InputValues WHERE ID_Name = :ID'
  else if QueryType = 'NNNYPoint Estimate' then
    sSql := 'SELECT CCT, XValue FROM InputValues WHERE ID_Name = :ID'
  else
    raise Exception.Create('Invalid QueryType value: ' + QueryType);

  Result := sSql;
end;

function TDataElement.GetValue(vType: string; iSystemSize, iSourceWater, iLSL,
  iCCT: integer): double;
begin
  Result := -9999;

  if ArrayType = 'NNNN' then
  begin
    if vType = 'PointEstimate' then
      Result := vXValue
    else if vType = 'Minimum' then
      Result := vMinValue
    else if vType = 'Maximum' then
      Result := vMaxValue
    else if vType = 'MostLikely' then
      Result := vMostLikely;
  end
  else if ArrayType = 'YNYY' then
  begin
    if vType = 'PointEstimate' then
      Result := YNYYPointEstimate[iSystemSize, iLSL, iCCT]
    else if vType = 'Minimum' then
      Result := YNYYMinimum[iSystemSize, iLSL, iCCT]
    else if vType = 'Maximum' then
      Result := YNYYMaximum[iSystemSize, iLSL, iCCT]
    else if vType = 'MostLikely' then
      Result := YNYYMostLikely[iSystemSize, iLSL, iCCT];
  end
  else if ArrayType = 'YNYN' then
  begin
    if vType = 'PointEstimate' then
```

```
    Result := YNYNPointEstimate[iSystemSize, iLSL]
  else if vType = 'Minimum' then
    Result := YNYNMinimum[iSystemSize, iLSL]
  else if vType = 'Maximum' then
    Result := YNYNMaximum[iSystemSize, iLSL]
  else if vType = 'MostLikely' then
    Result := YNYNMostLikely[iSystemSize, iLSL];
end
else if ArrayType = 'YNNY' then
begin
  if vType = 'PointEstimate' then
    Result := YNNYPointEstimate[iSystemSize, iCCT]
  else if vType = 'Minimum' then
    Result := YNNYMinimum[iSystemSize, iCCT]
  else if vType = 'Maximum' then
    Result := YNNYMaximum[iSystemSize, iCCT]
  else if vType = 'MostLikely' then
    Result := YNNYMostLikely[iSystemSize, iCCT];
end
else if ArrayType = 'YNNN' then
begin
  if vType = 'PointEstimate' then
    Result := YNNNPointEstimate[iSystemSize]
  else if vType = 'Minimum' then
    Result := YNNNMinimum[iSystemSize]
  else if vType = 'Maximum' then
    Result := YNNNMaximum[iSystemSize]
  else if vType = 'MostLikely' then
    Result := YNNNMostLikely[iSystemSize];
end
else if ArrayType = 'YYYN' then
begin
  if vType = 'PointEstimate' then
    Result := YYYNPointEstimate[iSystemSize, iSourceWater, iLSL]
  else if vType = 'Minimum' then
    Result := YYYNMinimum[iSystemSize, iSourceWater, iLSL]
  else if vType = 'Maximum' then
    Result := YYYNMaximum[iSystemSize, iSourceWater, iLSL]
  else if vType = 'MostLikely' then
    Result := YYYNMostLikely[iSystemSize, iSourceWater, iLSL];
end
else if ArrayType = 'YYNY' then
begin
  if vType = 'PointEstimate' then
    Result := YYNYPointEstimate[iSystemSize, iSourceWater, iCCT]
  else if vType = 'Minimum' then
    Result := YYNYMinimum[iSystemSize, iSourceWater, iCCT]
  else if vType = 'Maximum' then
    Result := YYNYMaximum[iSystemSize, iSourceWater, iCCT]
  else if vType = 'MostLikely' then
```

```
    Result := YYNYMostLikely[iSystemSize, iSourceWater, iCCT];
end
else if ArrayType = 'YYYY' then
begin
  if vType = 'PointEstimate' then
    Result := YYYYPointEstimate[iSystemSize, iSourceWater, iLSL, iCCT]
  else if vType = 'Minimum' then
    Result := YYYYMinimum[iSystemSize, iSourceWater, iLSL, iCCT]
  else if vType = 'Maximum' then
    Result := YYYYMaximum[iSystemSize, iSourceWater, iLSL, iCCT]
  else if vType = 'MostLikely' then
    Result := YYYYMostLikely[iSystemSize, iSourceWater, iLSL, iCCT];
end
else if ArrayType = 'NYYN' then
begin
  if vType = 'PointEstimate' then
    Result := NYYNPointEstimate[iSourceWater, iLSL]
  else if vType = 'Minimum' then
    Result := NYYNMinimum[iSourceWater, iLSL]
  else if vType = 'Maximum' then
    Result := NYYNMaximum[iSourceWater, iLSL]
  else if vType = 'MostLikely' then
    Result := NYYNMostLikely[iSourceWater, iLSL];
end
else if ArrayType = 'NNYN' then
begin
  if vType = 'PointEstimate' then
    Result := NNYNPointEstimate[iLSL]
  else if vType = 'Minimum' then
    Result := NNYNMinimum[iLSL]
  else if vType = 'Maximum' then
    Result := NNYNMaximum[iLSL]
  else if vType = 'MostLikely' then
    Result := NNYNMostLikely[iLSL];
end
else if ArrayType = 'NYNN' then
begin
  if vType = 'PointEstimate' then
    Result := NYNNPointEstimate[iSourceWater]
  else if vType = 'Minimum' then
    Result := NYNNMinimum[iSourceWater]
  else if vType = 'Maximum' then
    Result := NYNNMaximum[iSourceWater]
  else if vType = 'MostLikely' then
    Result := NYNNMostLikely[iSourceWater];
end
else if ArrayType = 'YYNN' then
begin
  if vType = 'PointEstimate' then
    Result := YYNNPointEstimate[iSystemSize, iSourceWater];
```

```
  end
  else if ArrayType = 'NYNY' then
  begin
    if vType = 'PointEstimate' then
      Result := NYNYPointEstimate[iSourceWater, iCCT]
    else if vType = 'Minimum' then
      Result := NYNYMinimum[iSourceWater, iCCT]
    else if vType = 'Maximum' then
      Result := NYNYMaximum[iSourceWater, iCCT]
    else if vType = 'MostLikely' then
      Result := NYNYMostLikely[iSourceWater, iCCT];
  end
  else if ArrayType = 'NNNY' then
  begin
    if vType = 'PointEstimate' then
      Result := NNNYPointEstimate[iLSL]
    else if vType = 'Minimum' then
      Result := NNNYMinimum[iLSL]
    else if vType = 'Maximum' then
      Result := NNNYMaximum[iLSL]
    else if vType = 'MostLikely' then
      Result := NNNYMostLikely[iLSL];
  end
  else
    raise Exception.Create('Invalid ArrayType value: ' + ArrayType);

end;

procedure TDataElement.ReadData;
begin
  ExecQuery;
end;

procedure TDataElement.SetQueryType;
begin
  QueryType := IfThen(strata_system_size, 'Y', 'N');
  QueryType := QueryType + IfThen(strata_source_water, 'Y', 'N');
  QueryType := QueryType + IfThen(strata_lsl, 'Y', 'N');
  QueryType := QueryType + IfThen(strata_cct, 'Y', 'N');
  QueryType := QueryType + TDistributionTypeToStr(distribution);

  ArrayType := copy(QueryType,1,4);
  StrataType := ArrayType;
end;

{ TDataElements }

procedure TDataElements.Add(key: string; de: TDataElement);
begin
  de.FAccessDB := FAccessDB;
```

```
  de.FAccessQry := FAccessQry;
  de.ReadData;

  DataStore.Add(key, de);
end;

function TDataElements.BaselineSame(varname: string): boolean;
var
  sSql: string;
begin
  sSql := 'SELECT NDWACSame FROM InputDesc WHERE ID_Name = :ID';

  FAccessQry.SQL.Clear;
  FAccessQry.SQL.Add(sSql);
  //FAccessQry.ParamByName('ID').AsString := varname;
  FAccessQry.Parameters.ParamByName('ID').Value := varname;
  FAccessQry.Open;
  if not FAccessQry.Eof then
  begin
    if FAccessQry.FieldByName('NDWACSame').AsString = 'Y' then
      Result := true
    else
      Result := false;
  end
  else
    Result := false;

  FAccessQry.Close;
end;

constructor TDataElements.Create(ADataPath: string);
begin
  DataPath := ADataPath;

  DataStore := TObjectDictionary<string, TDataElement>.Create([doOwnsValues]);
  FAccessDB := TADOConnection.Create(nil);
  FAccessDB.CursorLocation:=clUseServer;
  FAccessDB.LoginPrompt := False;
  FAccessDB.Mode:=cmRead;

  FAccessDB.ConnectionString:=format(ADOConStr,[DataPath]);
  FAccessDB.Open;

  FAccessQry := TADOQuery.Create(nil);
  FAccessQry.Connection := FAccessDB;
end;

destructor TDataElements.Destroy;
begin
  FAccessDB.Close;
```

```
  FAccessQry.Free;
  FAccessDB.Free;

  DataStore.Free;

  inherited;
end;

function TDataElements.GetVariable(varname: string): TDataElement;
begin
  try
    Result := DataStore.Items[varname];
  except
    on E: Exception do
    begin
      Result := nil;
    end;
  end;
end;

end.
```