

DataElements.pas

```
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: 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;
```

DataElements.pas

```
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;

YNYYPPointEstimate: TDoubleArray3;
YNYYPMinimum: TDoubleArray3;
YNYYPMaximum: TDoubleArray3;
YNYYPMostLikely: TDoubleArray3;
YNYYPMean: TDoubleArray3;
YNYYPStdDev: 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;

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

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

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

DataElements.pas

```
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
    YYYYPoint Estimate
    YYYYTriangular
    YYYYUniform
    YNYYPoint Estimate
    YNYYPoint Estimate
    YNYYUniform
    YNYYTriangular
    YNYYPoint Estimate
    YNYYTriangular
    YNYYUniform
    YNNYPoint Estimate
    YNNYUniform
    YYYYPoint Estimate
    YYYYTriangular
    YYYYUniform
    NYYNPoint Estimate
    NYYNTriangular
    NNYNPoint Estimate
    NNYNTriangular
    NYNNPoint Estimate
    NYNNUniform
    NYNYPoint Estimate
    NNNYPoint Estimate

  ArrayType:
    NNNN
    YNYY
    YNYY
    YNYY
    YNNN
    YYYY
    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 = 'YNNPoint Estimate' then
begin
  SetLength(YNNPointEstimate, 9, 3);
end
else if QueryType = 'YNYPoint Estimate' then
begin
  SetLength(YNYPointEstimate, 9, 3, 2);
end
else if QueryType = 'YNNFunction' 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 = 'YYNPoint Estimate' then
begin
  SetLength(YYNPointEstimate, 9, 3, 2);
end
else if QueryType = 'YYNTriangular' then
begin
  SetLength(YYNMinimum, 9, 3, 2);
  SetLength(YYNMaximum, 9, 3, 2);
  SetLength(YYNMostLikely, 9, 3, 2);
end
end

```

```
else if QueryType = 'YYNUniform' then
begin
  SetLength(YYNMinimum, 9, 3, 2);
  SetLength(YYNMaximum, 9, 3, 2);
end
else if QueryType = 'YNYPoint Estimate' then
begin
  SetLength(YNYPointEstimate, 9, 2, 2);
end
else if QueryType = 'YNYUniform' then
begin
  SetLength(YNYMinimum, 9, 2, 2);
  SetLength(YNYMaximum, 9, 2, 2);
end
else if QueryType = 'YNYTriangular' then
begin
  SetLength(YNYMinimum, 9, 2, 2);
  SetLength(YNYMaximum, 9, 2, 2);
  SetLength(YNYMostLikely, 9, 2, 2);
end
else if QueryType = 'YNYPoint Estimate' then
begin
  SetLength(YNYPointEstimate, 9, 2);
end
else if QueryType = 'YNYTriangular' then
begin
  SetLength(YNYMinimum, 9, 2);
  SetLength(YNYMaximum, 9, 2);
  SetLength(YNYMostLikely, 9, 2);
end
else if QueryType = 'YNYUniform' then
begin
  SetLength(YNYMinimum, 9, 2);
  SetLength(YNYMaximum, 9, 2);
end
else if QueryType = 'YNYPoint Estimate' then
begin
  SetLength(YNYPointEstimate, 9, 2);
end
else if QueryType = 'YNYUniform' then
begin
  SetLength(YNYMinimum, 9, 2);
  SetLength(YNYMaximum, 9, 2);
end
else if QueryType = 'YYYPoint Estimate' then
begin
  SetLength(YYYPointEstimate, 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 = 'NYYNPoint Estimate' then
begin
  SetLength(NYYNPointEstimate, 3, 2);
end
else if QueryType = 'NNNYPoint Estimate' then
begin
  SetLength(NNNYPointEstimate, 2);
end
```


DataElements.pas

```
else
raise Exception.Create('Invalid QueryType value: ' + QueryType);

querySQL := GetQuerySQL;

HasData := false;
end;

destructor TDataElement.Destroy;
begin
  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
```

```

                                DataElements.pas
    vFunction := FAccessQry.FieldName('Function').AsString;
end
else if QueryType = 'YNNNPoint Estimate' then
begin
    YNNNPointEstimate[FAccessQry.FieldName('SystemSize').AsInteger] :=
FAccessQry.FieldName('XValue').AsFloat;
end
else if QueryType = 'YNNNTriangular' then
begin
    YNNNMinimum[FAccessQry.FieldName('SystemSize').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
    YNNNMaximum[FAccessQry.FieldName('SystemSize').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
    YNNNMostLikely[FAccessQry.FieldName('SystemSize').AsInteger] :=
FAccessQry.FieldName('MostLikely').AsFloat;
end
else if QueryType = 'YNNUniform' then
begin
    YNNNMinimum[FAccessQry.FieldName('SystemSize').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
    YNNNMaximum[FAccessQry.FieldName('SystemSize').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
end
else if QueryType = 'YYYNPoint Estimate' then
begin
    YYYNPointEstimate[FAccessQry.FieldName('SystemSize').AsInteger,
                        FAccessQry.FieldName('SourceWater').AsInteger,
                        FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('XValue').AsFloat;
end
else if QueryType = 'YYYNTriangular' then
begin
    YYYNMinimum[FAccessQry.FieldName('SystemSize').AsInteger,
                FAccessQry.FieldName('SourceWater').AsInteger,
                FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
    YYYNMaximum[FAccessQry.FieldName('SystemSize').AsInteger,
                FAccessQry.FieldName('SourceWater').AsInteger,
                FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
    YYYNMostLikely[FAccessQry.FieldName('SystemSize').AsInteger,
                   FAccessQry.FieldName('SourceWater').AsInteger,
                   FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MostLikely').AsFloat;
end
else if QueryType = 'YYYNUniform' then
begin
    YYYNMinimum[FAccessQry.FieldName('SystemSize').AsInteger,

```

```

                                DataElements.pas
                                FAccessQry.FieldName('SourceWater').AsInteger,
                                FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
    YYNMaximum[FAccessQry.FieldName('SystemSize').AsInteger,
                FAccessQry.FieldName('SourceWater').AsInteger,
                FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
    end
    else if QueryType = 'YNYPoint Estimate' then
    begin
        YNYPointEstimate[FAccessQry.FieldName('SystemSize').AsInteger,
                           FAccessQry.FieldName('SourceWater').AsInteger,
                           FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('XValue').AsFloat;
    end
    else if QueryType = 'YNYPoint Estimate' then
    begin
        YNYPointEstimate[FAccessQry.FieldName('SystemSize').AsInteger,
                           FAccessQry.FieldName('LSL').AsInteger,
                           FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('XValue').AsFloat;
    end
    else if QueryType = 'YNYUniform' then
    begin
        YNYMinimum[FAccessQry.FieldName('SystemSize').AsInteger,
                    FAccessQry.FieldName('LSL').AsInteger,
                    FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
        YNYMaximum[FAccessQry.FieldName('SystemSize').AsInteger,
                    FAccessQry.FieldName('LSL').AsInteger,
                    FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
    end
    else if QueryType = 'YNYTriangular' then
    begin
        YNYMinimum[FAccessQry.FieldName('SystemSize').AsInteger,
                    FAccessQry.FieldName('LSL').AsInteger,
                    FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
        YNYMaximum[FAccessQry.FieldName('SystemSize').AsInteger,
                    FAccessQry.FieldName('LSL').AsInteger,
                    FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
        YNYMostLikely[FAccessQry.FieldName('SystemSize').AsInteger,
                       FAccessQry.FieldName('LSL').AsInteger,
                       FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MostLikely').AsFloat;
    end

```

```

                                DataElements.pas
else if QueryType = 'YNYNPoint Estimate' then
begin
    YNYNPointEstimate[FAccessQry.FieldName('SystemSize').AsInteger,
                      FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('XValue').AsFloat;
end
else if QueryType = 'YNYNTriangular' then
begin
    YNYNMinimum[FAccessQry.FieldName('SystemSize').AsInteger,
                FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
    YNYNMaximum[FAccessQry.FieldName('SystemSize').AsInteger,
                FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
    YNYNMostLikely[FAccessQry.FieldName('SystemSize').AsInteger,
                   FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MostLikely').AsFloat;
end
else if QueryType = 'YNYNUniform' then
begin
    YNYNMinimum[FAccessQry.FieldName('SystemSize').AsInteger,
                FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
    YNYNMaximum[FAccessQry.FieldName('SystemSize').AsInteger,
                FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
end
else if QueryType = 'YNNYPoint Estimate' then
begin
    YNNYPointEstimate[FAccessQry.FieldName('SystemSize').AsInteger,
                      FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('XValue').AsFloat;
end
else if QueryType = 'YNNYUniform' then
begin
    YNNYMinimum[FAccessQry.FieldName('SystemSize').AsInteger,
                FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
    YNNYMaximum[FAccessQry.FieldName('SystemSize').AsInteger,
                FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
end
else if QueryType = 'YYYYPoint Estimate' then
begin
    YYYYPointEstimate[FAccessQry.FieldName('SystemSize').AsInteger,
                      FAccessQry.FieldName('SourceWater').AsInteger,
                      FAccessQry.FieldName('LSL').AsInteger,
                      FAccessQry.FieldName('CCT').AsInteger] :=

```

```

FAccessQry.FieldName('XValue').AsFloat;
end
else if QueryType = 'YYYYTriangular' then
begin
    YYYYMinimum[FAccessQry.FieldName('SystemSize').AsInteger,
        FAccessQry.FieldName('SourceWater').AsInteger,
        FAccessQry.FieldName('LSL').AsInteger,
        FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
    YYYYMaximum[FAccessQry.FieldName('SystemSize').AsInteger,
        FAccessQry.FieldName('SourceWater').AsInteger,
        FAccessQry.FieldName('LSL').AsInteger,
        FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
    YYYYMostLikely[FAccessQry.FieldName('SystemSize').AsInteger,
        FAccessQry.FieldName('SourceWater').AsInteger,
        FAccessQry.FieldName('LSL').AsInteger,
        FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MostLikely').AsFloat;
end
else if QueryType = 'YYYYUniform' then
begin
    YYYYMinimum[FAccessQry.FieldName('SystemSize').AsInteger,
        FAccessQry.FieldName('SourceWater').AsInteger,
        FAccessQry.FieldName('LSL').AsInteger,
        FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
    YYYYMaximum[FAccessQry.FieldName('SystemSize').AsInteger,
        FAccessQry.FieldName('SourceWater').AsInteger,
        FAccessQry.FieldName('LSL').AsInteger,
        FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
end
else if QueryType = 'NYYNPointEstimate' then
begin
    NYYNPointEstimate[FAccessQry.FieldName('SourceWater').AsInteger,
        FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('XValue').AsFloat;
end
else if QueryType = 'NYYNTriangular' then
begin
    NYYNMinimum[FAccessQry.FieldName('SourceWater').AsInteger,
        FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
    NYYNMaximum[FAccessQry.FieldName('SourceWater').AsInteger,
        FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
    NYYNMostLikely[FAccessQry.FieldName('SourceWater').AsInteger,

```

```

                                DataElements.pas
                                FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MostLikely').AsFloat;
    end
    else if QueryType = 'NNYNPoint Estimate' then
    begin
        NNYNPointEstimate[FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('XValue').AsFloat;
    end
    else if QueryType = 'NYNNPoint Estimate' then
    begin
        NYNNPointEstimate[FAccessQry.FieldName('SourceWater').AsInteger] :=
FAccessQry.FieldName('XValue').AsFloat;
    end
    else if QueryType = 'NYNNUniform' then
    begin
        NYNNMinimum[FAccessQry.FieldName('SourceWater').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
        NYNNMaximum[FAccessQry.FieldName('SourceWater').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
    end
    else if QueryType = 'NNYNTriangular' then
    begin
        NYNNMinimum[FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MinValue').AsFloat;
        NYNNMaximum[FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MaxValue').AsFloat;
        NYNNMostLikely[FAccessQry.FieldName('LSL').AsInteger] :=
FAccessQry.FieldName('MostLikely').AsFloat;
    end
    else if QueryType = 'NYYNPoint Estimate' then
    begin
        NYYNPointEstimate[FAccessQry.FieldName('SourceWater').AsInteger,
                           FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('XValue').AsFloat;
    end
    else if QueryType = 'NNNYPoint Estimate' then
    begin
        NNNYPointEstimate[FAccessQry.FieldName('CCT').AsInteger] :=
FAccessQry.FieldName('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 = 'YNNPoint Estimate' then

sSql := 'SELECT SystemSize, SourceWater, XValue FROM InputValues WHERE ID_Name = :ID'

else if QueryType = 'YNNFunction' 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 = 'YYNPoint Estimate' then

sSql := 'SELECT SystemSize, SourceWater, LSL, XValue FROM InputValues WHERE ID_Name = :ID'

else if QueryType = 'YYNPoint Estimate' then

sSql := 'SELECT SystemSize, SourceWater, CCT, XValue FROM InputValues WHERE ID_Name = :ID'

else if QueryType = 'YYNTriangular' then

sSql := 'SELECT SystemSize, SourceWater, LSL, MinValue, MaxValue, MostLikely FROM InputValues WHERE ID_Name = :ID'

else if QueryType = 'YYNUniform' then

sSql := 'SELECT SystemSize, SourceWater, LSL, MinValue, MaxValue FROM InputValues WHERE ID_Name = :ID'

else if QueryType = 'YNYPoint Estimate' then

sSql := 'SELECT SystemSize, LSL, CCT, XValue FROM InputValues WHERE ID_Name = :ID'

else if QueryType = 'YNYUniform' then

sSql := 'SELECT SystemSize, LSL, CCT, MinValue, MaxValue FROM InputValues WHERE ID_Name = :ID'

else if QueryType = 'YNYTriangular' then

sSql := 'SELECT SystemSize, LSL, CCT, MinValue, MaxValue, MostLikely FROM InputValues WHERE ID_Name = :ID'

```

                                DataElements.pas
    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 = 'NYYNPoint Estimate' then
        sSql := 'SELECT SourceWater, XValue FROM InputValues WHERE ID_Name = :ID'
    else if QueryType = 'NYYNUniform' 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 = 'NNYNPoint 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 = 'YNY' then
    begin
        if vType = 'PointEstimate' then
            Result := YNYPointEstimate[iSystemSize, iLSL, iCCT]
        else if vType = 'Minimum' then
            Result := YNYMinimum[iSystemSize, iLSL, iCCT]
        else if vType = 'Maximum' then
            Result := YNYMaximum[iSystemSize, iLSL, iCCT]
        else if vType = 'MostLikely' then
            Result := YNYMostLikely[iSystemSize, iLSL, iCCT];
        end
    else if ArrayType = 'YNY' then
    begin
        if vType = 'PointEstimate' then
            Result := YNYPointEstimate[iSystemSize, iLSL]
        else if vType = 'Minimum' then
            Result := YNYMinimum[iSystemSize, iLSL]
        else if vType = 'Maximum' then
            Result := YNYMaximum[iSystemSize, iLSL]
        else if vType = 'MostLikely' then
            Result := YNYMostLikely[iSystemSize, iLSL];
        end
    else if ArrayType = 'YNY' then
    begin
        if vType = 'PointEstimate' then
            Result := YNYPointEstimate[iSystemSize, iCCT]
        else if vType = 'Minimum' then
            Result := YNYMinimum[iSystemSize, iCCT]
        else if vType = 'Maximum' then
            Result := YNYMaximum[iSystemSize, iCCT]
        else if vType = 'MostLikely' then
            Result := YNYMostLikely[iSystemSize, iCCT];
        end
    end
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]

```

```

                                DataElements.pas
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 = 'NYYN' then
begin
    if vType = 'PointEstimate' then
        Result := NYYNPointEstimate[iSourceWater, iCCT]
    else if vType = 'Minimum' then
        Result := NYYNMinimum[iSourceWater, iCCT]
    else if vType = 'Maximum' then
        Result := NYYNMaximum[iSourceWater, iCCT]
    else if vType = 'MostLikely' then
        Result := NYYNMostLikely[iSourceWater, iCCT];
end
else if ArrayType = 'NNNY' then
begin
    if vType = 'PointEstimate' then

```

```

                                DataElements.pas
    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;

```

DataElements.pas

```
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
```

DataElements.pas

```
    Result := DataStore.Items[varname];  
except  
  on E: Exception do  
  begin  
    Result := nil;  
  end;  
end;  
end;  
end.  
end.
```