Thanks for the questions and for taking the time to see how we can use GenericSql !    Please don’t hesitate to ask me any questions on how to use it, or if you have any suggestions for improving it.

Michael

         Using GenericSql to read from database

**1st version of GenericSql examples …**

       status = (GenericStaticUtilities.ParseEnum<BusinessServicePolicyStatusEnum>(((object[])((new GenericSQL())

               .Execute("select TOP 1 FieldValue FROM [ISORating].[dbo].[BondRatePremium] where workitem = " + quoteNum + " and FieldName = 'Eligible Status' order by id desc", GenericSQL.DbAction.read)[0]))[0].ToString()))

                     GenericSQL sql = new GenericSQL();

                     List<SqlParameter> sqlParam = new List<SqlParameter>();

                     List<object> uwInfo;

                     string query = "SELECT Value FROM CPP.dbo.BondProductAttribute WHERE ProductId=@productId AND Name=@name";

                     sqlParam.Add(new SqlParameter("@productId", \_PIM.ProductId));

                     sqlParam.Add(new SqlParameter("@name", "UWEMail" + (EnvironmentType.LIVE.Equals(GlobalItems.CurrentEnvironment) ? EnvironmentType.LIVE.GetDescriptionAttributeValue() : EnvironmentType.DEV.GetDescriptionAttributeValue())));

                     uwInfo = sql.Execute(query, GenericSQL.DbAction.read, sqlParam);

                     // if there were emails found, then use these

                     if (uwInfo != null && uwInfo.Count > 0)

                     {

                           uwEmailAddr = "";

                           foreach (object[] uwe in uwInfo)

                            {

                                   if (uwe[0].ToString().Trim().Length != 0)

                                  {

                                         uwEmailAddr += (cnt > 0 ? ";" : "") + uwe[0].ToString().Trim();

                                  }

                           }

                     }

var result = (new GenericSQL()).Execute("select \* FROM [ISORating].[dbo].[BondRatePremium] where workitem = " + quoteNum + " and profileId = " + i.Id + " order by created desc", GenericSQL.DbAction.read);

                   string score = null;

                   string publicRecords = null;

                   if (result != null && result.Any())

                   {

                       for (int j = 0; j < result.Count; j++)

                       {

                           var array = (object[]) result[j];

                           for (int k = 0; k < array.Length; k++)

                           {

                               if (array[k].ToString() == "Credit Score")

                                   score = array[k + 1].ToString();

                               if (array[k].ToString() == "Public Records Count")

                                   publicRecords = array[k + 1].ToString();

                           }

                       }

                   }

**Example using Execute to read *-> I’m just looking for whether or not something is there***

bool status = ((new GenericSQL()).Execute("select \* FROM [ISORating].[dbo].[BondRatePremium] where workitem = " + quoteNum + " AND ProfileId = " + profileId + " AND FieldName = 'Credit Score'", GenericSQL.DbAction.read) != null);

                     if (status)

                     {

       baseReportURL = serviceUtilities.GetSCLookupValue(ScListName.TransUnionReportURL, ScLookupName.TransUnionReportURL) + "?R=15&RefID=" + quoteNum + profileId;

                     }

                     return status;

         WE NEED SOMETHING BETTER to read from database

              Reading from the database using Entities/Models that represent a table

**Simple Example using ModelRead**

             private class myObj

                    {

                      public string FieldName { get; set; }

                      public string FieldValue { get; set; }

                      public DateTime Created { get; set; }

                      public string BusinessAction { get; set; }

                    }

                    Call GenericSQL ModelRead

                    var modelReadResult = (new GenericSQL()).ModelRead<myObj>("[ISORating].[dbo].[BondRatePremium]", "where workitem = 163548 order by created");

                    example of going through the modelResultSet using linq foreach...

                    modelReadResult.ForEach(x =>

                    {

                       var ba = x.BusinessAction;

                       var fn = x.FieldName;

                       var fa = x.FieldValue;

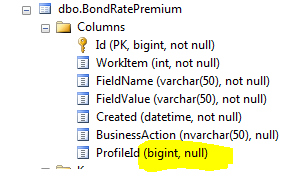
                       var  c = x.Created;

                    });

**Example of a custom Model from ISORating BondRatePremium Table**

!!! IMPORTANT !!!  **Add the** [GnSqlIgnoreOnInsert] **custom attribute for properties you don’t want part of an insert (inserts discuss later below), like primary key Id below that gets incremented automatically.**

**GenericSQL will try to insert all properities in your model, so be sure to use** [GnSqlIgnoreOnInsert]



namespace BondCommon.Models

{

                            public class BondRatePremium : IBondRatePremium

                            {

                                   [GnSqlIgnoreOnInsert]                          <- IgnoreOnInsert attribute added so GenericSQL won’t add this part of the Insert statement

                                   public int Id { get; set; }

                            public int WorkItem { get; set; }

                                   public string FieldName { get; set; }

                                  public string FieldValue { get; set; }

[GnSqlIgnoreOnInsert]

                                   public DateTime Created { get; set; }

                                   public string BusinessAction { get; set; }

                                   public long? ProfileId { get; set; }     <- make sure to add nullable types if the SQL column allows nulls

                            }

}

private List<string> GetReasons(int quoteNum)

                     {

                                                                               You don’t need to add a select here

var modelReadResult = (new GenericSQL()).ModelRead<BondRatePremium>("[ISORating].[dbo].[BondRatePremium]", "where workitem = " + quoteNum + " and BusinessAction = 'Valid' and FieldName like 'Reason%' ");

                            List<string> reasonList = null;

                            if (modelReadResult != null)

                            {

                                   reasonList = new List<string>();

                                   modelReadResult.ForEach(x => reasonList.Add(x.FieldValue));

                            }

                            return reasonList;

                     }

**Another Example using ModelRead**

var modelReadResult = (new GenericSQL()).ModelRead<BondRatePremium>("[ISORating].[dbo].[BondRatePremium]", "where workitem= " + quoteNum + " order by created");

                           vmEligibility.RatingResultsExist = (modelReadResult != null && modelReadResult.Count > 0);

                           if (modelReadResult != null)

                           {

                                  var bondPremiumAmount = modelReadResult.FirstOrDefault(x => x.FieldName == "Premium Amount" && x.BusinessAction == "Rate");

                                  if (bondPremiumAmount != null)

                                         if (String.IsNullOrWhiteSpace(bondPremiumAmount.FieldValue) || Convert.ToInt32(bondPremiumAmount.FieldValue) < 1)

                                                vmEligibility.BondPremium = String.Empty;

                                         else

                                                vmEligibility.BondPremium = string.Format("{0:C0}", Convert.ToInt32(bondPremiumAmount.FieldValue));

                                  else

                                         vmEligibility.BondPremium = String.Empty;

                           }

                           else

                                  vmEligibility.BondPremium = String.Empty;

**Another Example using ModelRead in BusinessServices**

                     public RateScoreCrossRef GetRateScoreCrossRef(string classCd, string rateType, string stateNo, int coId, int creditScore)

                     {

                            const string whereClause = @"WHERE ClassCD = @ClassCD

                                                                     AND RateType = @RateType

                                                                     AND StateNo = @StateNo

                                                                     AND CoID = @CoID

                                                                     AND LowScore <= @CreditScore AND @CreditScore <= HighScore";

                            var parms = new List<SqlParameter>(){

                                   new SqlParameter("@ClassCD", classCd),

                                   new SqlParameter("@RateType", rateType),

                                   new SqlParameter("@StateNo", stateNo),

                                  new SqlParameter("@CoID", coId),

                                  new SqlParameter("@CreditScore", creditScore)

                            };

(new GenericSQL(GenericSQL.Context.SuretyCentralContext)).ModelRead<RateScoreCrossRef>("SELECT TOP 1 \* FROM [SCBefore].[dbo].[RateScoreCrossRef]", whereClause, parms);

                            if (rscr == null)

                                   return null;

                            else

                                   return rscr.FirstOrDefault();

                     }

Reading from the database using Dynamic Models

                    C**reate your anonomous model of columns you want returned...**

                                                  db type        Column Name                 default value of the type -> you have to put the default value!

                     string    FieldName      =  String.Empty;

                     string    BusinessAction =  String.Empty;

                     int       WorkItem       =  0;

                     DateTime  Created        =  DateTime.Now;

                     var model = new { FieldName, BusinessAction, WorkItem, Created };

**Call GenericSQL DynamicRead**

                                                                                    You don’t need to add a select here

                     var dynamicReadResult = (new GenericSQL()).DynamicRead(model, "[ISORating].[dbo].[BondRatePremium]", "where workitem = 163548 order by created");

**Example of going throught the resultSet...**

                     foreach (var e in dynamicReadResult)

                     {

                           use GenericSQL extension methods ValueOf and ItemOf to get the data...

                           var baValue = e.ValueOf("BusinessAction");

                           var baItem = e.ItemOf("BusinessAction");

                           var workItemValue = e.ValueOf("WorkItem");

                           var workItem = e.ItemOf("WorkItem");

                           var createdValue = e.ValueOf("Created");

                           var createdItem = e.ItemOf("Created");

                     }

Using the Schematable from the Database to return a list of strongly typed models

**Call SchemaRead with your select statement** -> GenericSQL will read from the db schematable to get the types and return actual strong types for each value

                     var schemaReadResult = (new GenericSQL()).SchemaRead("SELECT TOP 1000 \* FROM [CPP].[dbo].[abProfile] where TaxIdNumber is not null");

**Example of going throught the resultSet...**

                     foreach (var e in schemaReadResult)

                     {

                           use GenericSQL extension methods ValueOf and ItemOf to get the data...

                           var baValue = e.ValueOf("TaxIdNumber");

                           var baItem  = e.ItemOf ("TaxIdNumber");

                           var createdValue = e.ValueOf("Created");

                           var createdItem  = e.ItemOf ("Created");

                     }

                           Deleting records from the database

**Simple Example**

                                int result = (new GenericSQL()).Execute(query, GenericSQL.DbAction.delete)  **->  a list of SqlParameters is optional for 3rd param not shown in this example**

**Example of it being used in BusinessCentralService.cs**

              public void DeleteBondRatePremium(int quoteNum, string businessAction, string fieldName = null)

              {

                     var sqlParam = new List<SqlParameter>();

                     sqlParam.Add(new SqlParameter("@quoteNum", quoteNum));

                     sqlParam.Add(new SqlParameter("@businessAction", businessAction));

                     string query = "DELETE FROM [ISORating].[dbo].[BondRatePremium] WHERE WorkItem = @quoteNum AND BusinessAction = @businessAction";

                     if (!string.IsNullOrEmpty(fieldName))

                     {

                           query += " AND FieldName LIKE '@fieldName%'";

                           sqlParam.Add(new SqlParameter("@fieldName", fieldName));

                     }

                     (new GenericSQL()).Execute(query, GenericSQL.DbAction.delete, sqlParam);

              }

              public int UpdateBondQuoteInforcePremium(int quoteNum, decimal inforcePremium)

              {

                     //(new GenericSQL()).Execute(query, GenericSQL.DbAction.delete, sqlParam);

                     //new GenericSQL(GenericSQL.Context.BusinessCentralContext).Execute("update CPP.dbo.bondQuote where ");

                     const string query = @"UPDATE [CPP].[dbo].[bondQuote] SET inforcePremium = @inforcePremium

                                                              WHERE quoteNum = @quoteNum";

                     var parms = new List<SqlParameter>(){

                           new SqlParameter("@quoteNum", quoteNum),

                           new SqlParameter("@inforcePremium", inforcePremium)

                     };

                     var rowCnt = (new GenericSQL(GenericSQL.Context.BusinessCentralContext)).ExecuteNonQuery(query, GenericSQL.DbAction.update, parms);

                     return rowCnt;

              }

**Example of deleting records using ExecuteNonQuery**

              [System.Web.Http.HttpPut]

              [System.Web.Http.Route("resetquote/")]   //[System.Web.Http.Route("{quoteNum:int:min(1)}")]

              public IHttpActionResult Put([FromBody]SesssionManager.PIM\_UserModel\_Combined pimUM)

              {

                     // delete rating records

                     if ((new GenericSQL()).ExecuteNonQuery("delete from [ISORating].[dbo].[BondRatePremium] where workitem = " + pimUM.QuoteNum) < 1)

                     {

                           //do something?

}

                     // change quote status back to Quote

                     Quote quote = \_quoteService.Find(Convert.ToInt32(pimUM.QuoteNum));

                     if (quote == null)

                     {

                           throw new Exception("quoteService could not find " + pimUM.QuoteNum);

                     }

                     quote.Status = PolicyStatusEnum.Quote.ToString();

                     \_quoteService.Update(quote);

                     int recordsWrittenNum = \_unitOfWork.SaveChanges();

                     if (recordsWrittenNum < 1)

                     {

                           throw new Exception("Roll back Quote Status to 'Quote' failed during ResetQuote for Quote Number " + pimUM.QuoteNum);

                     }

                     /////  Update/add AgentApplicationTransLog Activities ////////////////////////////////////////////////

                     var aaTrans = new AgentApplicationTransLogService();

                     // update previouse activities

                     aaTrans.**SetAllToInconsequential**(pimUM.AgentId, pimUM.QuoteNum);

                     // add new activity

                     var agentApplicationTransLog = CreateAgentApplicationTransLogModel(pimUM);

                     agentApplicationTransLog.Activity = "Quote Created";

                     agentApplicationTransLog.InconsequentialActivity = 0;

                     agentApplicationTransLog.RequestSource = "Approved Summary Page";

                     aaTrans.**Add**(agentApplicationTransLog);   <-  adding a new record here

                     return Ok(1);

              }

                Updating Records using GenericSql Services

**I created a base service for GenericSql to do table updates using the schema information from the database to verify column names and data types.**

**Below is an example of how to use it for the AgentApplicationTransLog table (***updating the InsName to the current Principal name***).**

              var aaTransLogSvc = new AgentApplicationTransLogService();

              aaTransLogSvc.ColumnsToUpdate.AddColumn("InsName", profilePrincipal.NameFull);

              aaTransLogSvc.FieldConditions.AddFieldCondition(SqlWord.WHERE, "InconsequentialActivity", SqlWord.Equals, "false");

              aaTransLogSvc.FieldConditions.AddFieldCondition(SqlWord.AND, "QuoteNum", SqlWord.Equals, profilePrincipal.QuoteNum.ToString());

              aaTransLogSvc.Update();

**Please note that all values (the last paramater of AddColumn and AddFieldCondition) datatypes:  int’s, date’s, bool, etc. must be put into a string like -> profilePrincipal.QuoteNum.ToString() above**

**Also, there is a property called FilterConditionString where you can add whatever where clause you want that over-rides the the FieldConditions collection above.**

                                              // this example of GenericSqlService does the exact same thing as the AgentApplicationTransLogService() (example below) only difference is, you can pass any DB/TableName into GenericSqlService

                                               var gnSQl = new GenericSqlService("[GrandCentral].[dbo].[AgentApplicationTransLog]");

                     gnSQl.ColumnsToUpdate.AddColumn("InsName", profilePrincipal.NameFull);

                     gnSQl.FieldConditions.AddFieldCondition(SqlWord.WHERE, "InconsequentialActivity", SqlWord.Equals, "false");

                     gnSQl.FieldConditions.AddFieldCondition(SqlWord.AND, "QuoteNum", SqlWord.Equals, profilePrincipal.QuoteNum.ToString());

                     gnSQl.Update();

                                                // does exact same thing as above;  AgentApplicationTransLogService inherits GenericSqlService

                     var aaTransLogSvc = new AgentApplicationTransLogService();

                     aaTransLogSvc.ColumnsToUpdate.AddColumn("InsName", profilePrincipal.NameFull);

                     aaTransLogSvc.FieldConditions.AddFieldCondition(SqlWord.WHERE, "InconsequentialActivity", SqlWord.Equals, "false");

                     aaTransLogSvc.FieldConditions.AddFieldCondition(SqlWord.AND, "QuoteNum", SqlWord.Equals, profilePrincipal.QuoteNum.ToString());

                     aaTransLogSvc.Update();

       (new AgentApplicationTransLogService()).SetAllToInconsequential(pimUM.AgentId, pimUM.QuoteNum);

Inserting Records using GenericSql

       public bool WriteCreditReport(CCKTrans cckTrans)

       {

              bool result = false;

              if ((int)(new GenericSQL(GenericSQL.Context.SuretyDBContext)).Execute<CCKTrans>(cckTrans, "[SuretyDB].[dbo].[CCKTrans]", GenericSQL.DbAction.insert) < 1)

                           throw new SqlExecutionException("CCKTrans INSERT FAILURE: GenericSQL()).Execute<CCKTrans> ");

              result = true;

              return result;

       }

       AgentApplicationTransLog agentApplicationTransLog = new AgentApplicationTransLog()

       {

                     LOB = "CPP",

                     EmployeeNum = userId,

                     AgtID = agentId,

                     AgtGroup = agtGrp,

                     Activity = null,

                     TransTimeStamp = DateTime.Now,

                     QuoteNum\_PolicyID = quoteNum,

                     NotificationSent = 1,

                     QuoteNum = quoteNum,

                     InconsequentialActivity = 1,

                     InsName = profile.NameFull

       };

       if ((int)(new GenericSQL()).Execute<IAgentApplicationTransLog>(agentApplicationTransLog, "[GrandCentral].[dbo].[AgentApplicationTransLog]", GenericSQL.DbAction.insert) < 1)

throw new SqlExecutionException("BOND APPROVED INSERT FAILURE: GenericSQL()).Execute<IAgentApplicationTransLog> --> rows returned less than 1 : agtid = " + pimUM.AgentId + " quoteNum = " + pimUM.QuoteNum);

       var bondRatePremium = new BondRatePremium()

       {

              WorkItem = quoteNum,

              FieldName = "Eligible Status",

              FieldValue = "Approve",

              Created = DateTime.Now,

              BusinessAction = "Eligible",

              ProfileId = profileId

       };

if ((int)(new GenericSQL()).Execute<IBondRatePremium>(bondRatePremium, "[ISORating].[dbo].[BondRatePremium]", GenericSQL.DbAction.insert) < 1)

                                                throw new SqlExecutionException("BondRatePremium INSERT FAILURE: GenericSQL()).Execute<IBondRatePremium> --> rows returned less than 1 : agtid = " + pimUM.AgentId + " quoteNum = " + pimUM.QuoteNum);