Get Oledb Table , Column and Fkeys

<https://docs.microsoft.com/en-us/iis/extensions/database-manager-reference/foreignkey-fktableschema-property-microsoft-web-management-databasemanager>

// Retrieve the information about a table.

public TableInfo GetTableInfo(string connectionString, string tableName, string schema)

{

// Create a new TableInfo object.

TableInfo info = new TableInfo();

// Create a connection to the database.

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

connection.Open();

// Retrieve the table name.

info.Name = tableName;

// Retrieve the list of table columns.

GetColumns(connection, tableName, schema, info.Columns);

// Retrieve the list of the table's foreign keys.

GetForeignKeys(connection, tableName, info.ForeignKeys);

// Retrieve the list of the table's indices.

GetIndexes(connection, tableName, info.Indexes);

}

// Return the table information.

return info;

}

// Retrieve the list of table colums.

private void GetColumns(OleDbConnection connection, string tableName, string tableSchema, ICollection<Column> columns)

{

String[] restrictions = new string[] { null, null, tableName, null };

DataTable columnSchema;

// Retrieve a list of primary keys for a table.

string[] primaryKeys = GetPrimaryKey(

connection.ConnectionString, tableName, tableSchema);

// Open the schema information for the columns.

columnSchema = connection.GetSchema(

OleDbMetaDataCollectionNames.Columns, restrictions);

// Enumerate the table's columns.

foreach (DataRow row in columnSchema.Rows)

{

// Create a new column object.

Column column = new Column();

// Retrieve the column name.

column.Name = row["COLUMN\_NAME"].ToString();

column.OriginalName = row["COLUMN\_NAME"].ToString();

// Test if the column allows nulls.

column.AllowNulls = (bool)row["IS\_NULLABLE"];

// Retrieve the column type.

column.TypeInfo = GetColumnType(connection,

OleDbTypeToString((OleDbType)

Int32.Parse(row["DATA\_TYPE"].ToString())));

// Retrieve the column's default value.

column.DefaultValue = ((row["COLUMN\_DEFAULT"] as DBNull)

!= null) ? "Null" : row["COLUMN\_DEFAULT"].ToString();

// Retrieve the column's precision.

column.Precision = ((row["NUMERIC\_PRECISION"] as DBNull)

!= null) ? 0 : Int32.Parse(row["NUMERIC\_PRECISION"].ToString());

// Retrieve the column's scale.

column.Scale = ((row["NUMERIC\_SCALE"] as DBNull) != null) ? 0 : Int32.Parse(row["NUMERIC\_SCALE"].ToString());

// Specify if the column is a primary key.

column.InPrimaryKey = primaryKeys.Contains(row["COLUMN\_NAME"].ToString());

// Specify that the column is not an identity column.

column.IsIdentity = false;

// Retrieve the column length.

column.Length = ((OleDbType)Int32.Parse(row["DATA\_TYPE"].ToString()) != OleDbType.WChar) ? -1 : Int32.Parse(row["CHARACTER\_MAXIMUM\_LENGTH"].ToString());

// Append the column to the list.

columns.Add(column);

}

}

// Determine the type of a column.

private TypeInfo GetColumnType(OleDbConnection connection, string columnName)

{

// Retrieve the list of column types.

ICollection<TypeInfo> serverTypes = GetServerTypes(connection.ConnectionString);

// Enumerate the columns.

foreach (TypeInfo type in serverTypes)

{

// Return the column type if it is found.

if (type.Name.Equals(columnName))

{

return type;

}

}

// Raise an exception if the column type was not found.

throw new ArgumentException("Unknown column type: " + columnName);

}

// Retrieve the list of a table's indices.

private void GetIndexes(OleDbConnection connection, string tableName, IList<Index> indices)

{

String[] restrictions = new string[] { null, null, null, null, tableName };

DataTable schema;

// Open the schema information for the indices.

schema = connection.GetOleDbSchemaTable(OleDbSchemaGuid.Indexes, restrictions);

// Enumerate the table's indices.

foreach (DataRow row in schema.Rows)

{

// Create a new index.

Index dbIndex = new Index();

// Append the index name.

dbIndex.Name = row["INDEX\_NAME"].ToString();

dbIndex.OriginalName = row["INDEX\_NAME"].ToString();

// Append the index's uniqueness.

dbIndex.Unique = (bool)row["UNIQUE"];

// Specify the index type.

dbIndex.IndexType = (bool)row["PRIMARY\_KEY"] == true ? IndexType.PrimaryKey : IndexType.Index;

// Create an index column object.

IndexColumn column = new IndexColumn();

column.Name = row["COLUMN\_NAME"].ToString();

// Specify whether the index is descending.

column.Descending = (Int32.Parse(row["COLLATION"].ToString()) == 2) ? true : false;

dbIndex.Columns.Add(column);

// Append the index to the list.

indices.Add(dbIndex);

}

}

// Retrieve the list of a table's foreign keys.

private void GetForeignKeys(OleDbConnection connection, string tableName, IList<ForeignKey> foreignKeys)

{

String[] restrictions = new string[] { null };

DataTable schema;

// Open the schema information for the foreign keys.

schema = connection.GetOleDbSchemaTable(OleDbSchemaGuid.Foreign\_Keys, restrictions);

// Enumerate the table's foreign keys.

foreach (DataRow row in schema.Rows)

{

ForeignKey dbForeignKey = new ForeignKey();

dbForeignKey.Name = row["FK\_NAME"].ToString();

dbForeignKey.OriginalName = row["FK\_NAME"].ToString();

dbForeignKey.FKTableName = row["FK\_TABLE\_NAME"].ToString();

ForeignKeyColumn fkc = new ForeignKeyColumn();

fkc.Name = row["FK\_COLUMN\_NAME"].ToString();

dbForeignKey.FKColumns.Add(fkc);

dbForeignKey.FKTableSchema = schema.ToString();

dbForeignKey.PKTableName = row["PK\_TABLE\_NAME"].ToString();

ForeignKeyColumn pkc = new ForeignKeyColumn();

pkc.Name = row["PK\_COLUMN\_NAME"].ToString();

dbForeignKey.PKColumns.Add(pkc);

dbForeignKey.PKTableSchema = schema.ToString();

foreignKeys.Add(dbForeignKey);

}

}

...

// Map OLEDB data types to their respective string descriptions.

internal static string OleDbTypeToString(OleDbType type)

{

switch (type)

{

case OleDbType.Binary:

return "Binary";

case OleDbType.Boolean:

return "Bit";

case OleDbType.UnsignedTinyInt:

return "Byte";

case OleDbType.TinyInt:

return "TinyInt";

case OleDbType.Integer:

return "Integer";

case OleDbType.Currency:

return "Currency";

case OleDbType.Date:

return "DateTime";

case OleDbType.Double:

return "Float";

case OleDbType.Guid:

return "UniqueIdentifier";

case OleDbType.Char:

case OleDbType.WChar:

return "Text";

case OleDbType.Single:

return "Real";

case OleDbType.SmallInt:

return "SmallInt";

case OleDbType.Numeric:

case OleDbType.Decimal:

return "Decimal";

default:

return "Unknown";

}

}