

Analysis Process Modeling Notation For Business Intelligence  
  
Technische Dokumentation

Eingereicht von

**Alexander Amstler**

**Philip Hohendanner**

**Maria Limberger**

Angefertigt am

**Institut für Wirtschaftsinformatik –**

**Data & Knowledge Engineering**

Beurteiler / Beurteilerin

**Dipl.-Ing. Thomas Neuböck**

***Univ.-Prof. Dipl.-Ing. Dr. techn. Michael Schrefl***

Projektstudie

Inhaltsverzeichnis

[1. Installationsanleitung 3](#_Toc507436488)

[2. Beschreibung der Projektmappe 4](#_Toc507436489)

[2.1. Database 4](#_Toc507436490)

[2.1.1. DBContext 4](#_Toc507436491)

[2.1.2. DBService 5](#_Toc507436492)

[2.2. View 8](#_Toc507436493)

[2.2.1. BaseMenu 8](#_Toc507436494)

[2.2.2. ComboItem 10](#_Toc507436495)

[2.2.3. CreateNewAnalysis 10](#_Toc507436496)

[2.2.4. Insert\_item 11](#_Toc507436497)

[2.2.5. LoadForm 11](#_Toc507436498)

[2.2.6. Login 11](#_Toc507436499)

[2.2.7. SaveSchema 13](#_Toc507436500)

[2.2.8. SelectNavigationOperator 13](#_Toc507436501)

[2.2.9. SelectTable 16](#_Toc507436502)

[2.2.10. UserInput 16](#_Toc507436503)

[2.3. Program 16](#_Toc507436504)

[3. Arbeitszeiten 17](#_Toc507436505)

[4. Abbildungsverzeichnis 18](#_Toc507436506)

# Installationsanleitung

Um an dem Projekt zu öffnen und weiterarbeiten zu können sind einige kurze Schritte zu erledigen welche im Folgenden kurz erklärt werden.

* 1. Visual Studio installieren

Als Programmiersprache wählten wir C# und verwendeten die Integrierte Entwicklungsumgebung Visual Studio 2017. Diese kann man unter der folgenden Webadresse downloaden: <https://www.visualstudio.com/de/>. Danach muss das Programm nur noch durch den Visual Studio Installer installiert werden, bevor man die Projektmappe öffnen kann. Bei der Installation ist darauf zu achten, „Entwicklung für die universelle Windows-Plattform“ und „.NET Desktopentwicklung“ anzukreuzen und mit zu installieren.

* 1. Projektmappe öffnen

Um eine Projektmappe zu öffnen klickt man im Menü auf Datei > Öffnen > Projekt/Projektmappe…. Dann wählt man den Speicherort der Projektmappe aus, öffnet den Ordner „OLAP\_WindowsForms“ und öffnet dann die Datei „OLAP\_WindowsForms.sln“. Es sollte sich nun das Projekt mitsamt allen benötigten Verweisen öffnen.

* 1. Verweise überprüfen/hinzufügen

Sollte das Projekt beim Starten Fehler enthalten, fehlen möglicherweise einige Verweise. Folgende Verweise sollten in dem Projekt vorhanden sein:

* Microsoft.CSharp
* Npgsql
* System
* System.Core
* System.Data
* System.Data.DataSetExtensions
* System.Deployment
* System.Drawing
* System.Net.Http
* System.Threading.Tasks.Extensions
* System.Windows.Forms
* System.Xml
* System.Xml.Linq

Sollte einer oder mehrere dieser Verweise nicht vorhanden sein, sind folgende Schritte auszuführen:

1. Im Menü ist folgendes auszuwählen: Projekt > NuGet-Pakete verwalten…
2. Es öffnet sich ein Fenster, in dem man links oben auf „Durchsuchen“ klickt.
3. In der Suchleiste werden die fehlenden Verweise eingegeben, ausgewählt und dann auf der rechten Seite auf Installieren gedrückt.

Wenn die oben angeführten Schritte ausgeführt wurden, sollte das Programm problemlos laufen.

# Beschreibung der Projektmappe

Die Projektmappe besteht grundsätzlich aus Windows Forms Komponenten und einfachen Klassen. Die Program Klasse besitzt eine Main Methode, mit der das Programm gestartet werden kann. Die restlichen Klassen sind in zwei Ordner unterteilt: Database und View. Alle Klassen sowie deren Felder und Methoden werden in den folgenden Kapiteln nach deren alphabethischer Reihenfolge, so wie sie auch in der Projektmappe zu finden sind, beschrieben.

# Database

In dem Ordner Database befinden sich die Klassen, welche den Zugriff auf eine PostgreSQL Datenbank ermöglichen. Diese Klassen sind:

* DBContext
* DBService

Die oben aufgelisteten Klassen werden in den folgenden Kapiteln mit deren Feldern und Methoden genauer beschrieben.

# DBContext

Die Klasse DBContext ermöglicht den Zugriff auf die Klasse DBService:

* \_service speichert den aktuellen Service als DBService

Mittels Initialize wird ein neuer DBService erstellt und in \_service abgespeichert.

Die Funktion Service gibt den abgespeicherten \_service zurück, falls dieser nicht null ist.

# DBService

Die Klasse DBService speichert sich alle wichtigen Login-Daten, um auf die PostgreSQL Datenbank zugreifen zu können:

* DBHostname speichert den Servernamen als string
* DBPort speichert die Portnummer als string
* DBUsername speichert den Usernamen als string
* DBPasswort speichert das Passwort des Users als string
* DBName speichert den Datenbanknamen als string
* NewTransatction speichert ob eine neue Transaktion gestartet wird oder ob es noch Teil einer alten Transaktion ist als bool
* nass\_dq\_sid speichert den Primary Key der Tabelle AGS\_NASS\_DIM\_QUAL als int
* nass\_sc\_sid speichert den Primary Key der Tabelle AGS\_NASS\_DIM\_QUAL\_SLICE\_COND als int
* ds speichert das aktuelle Daten-Set als DataSet
* dt speichert die aktuelle Daten-Tabelle als DataTable

Mittels dem LogIn werden sämtliche Daten (DBHostname, DBPort, DBUsername, DBPassword, DBName) für den Login für die Postgres Datenbank in den oben beschriebenen Attributen gespeichert.

Die folgenden Methoden sind speziell für die ComboItem Klasse gedacht:

GetData erstellt mittels NpgsqlConnection eine Verbindung zur PostgreSQL Datenbank und führt eine Abfrage, die als Input Parameter übergeben wird, aus und retourniert eine Tabelle als DataTable. Es gibt drei Überladungen der Methode, wobei diese:

* zwei Spalten einer Tabelle zurückliefert
* eine Spalte einer Tabelle zurückliefert
* als Input Parameter ein SQL Statement als string akzeptiert und eine Spalte einer Tabelle zurückgibt.

GetStringFormStmt ruft die Methode GetData auf und selektiert aus der zurückgegebenen Tabelle die Reihe und Spalte, welche als Parameter mitgegeben werden. Dann wird ein string zurückgegeben.

Die folgenden Methoden sind Hilfsmethoden:

GetLatestID nimmt die Parameter columnPk und table und liefert den int des größten Primary Key oder 1, wenn die Tabelle noch leer ist.

GetKeyfromTable nimmt die Parameter table, keyValue, keyColumn, targetColumn und liefert den int einer gewählten Spalte, wo der mitgegebene int Schlüssel dem Wert der mitgegebenen Spalte entspricht oder retourniert einen Error mit -1 falls nichts gefunden wird.

GetSKeyfromTable nimmt die Parameter table, keyValue, keyColumn, targetColumn und liefert den string einer gewählten Spalte, wo der mitgegebene int Schlüssel, dem Wert der mitgegebenen Spalte entspricht oder retourniert einen Error mit -1 falls nichts gefunden wird.

GetConnection gibt eine neue NpgsqlConnection zum PostgreSQL Server zurück.

Die folgenden Methoden sind für das Einfügen in Tabellen:

InsertInto holt sich den aktuellen Primary Key durch getLatestID und erstellt dann über StringBuilder ein SQL Insert-Statement. Danach weist es der übergebenen NpgsqlConnection und NpgsqlTrancation ein NpgsqlCommand zu, das aus dem SQL Insert-Statement und den mitgegebenen Insert\_items besteht.

InsertWithoutPK funktioniert ähnlich wie InsertInto, nur dass hier der Primary Key das erste Listenelement der Insert\_item Liste ist.

InsertDimQual ist für das Einfügen der Dimension Qualifications zuständig. Wenn es eine neue Transaktion ist werden einmalig die Primary Key Attribute mit den aktuellen Werten aus den Postgres Tabellen befüllt, ansonsten werden die bereits gespeicherten Primary Keys um 1 erhöht. Danach werden die Werte der jeweiligen ComboBox und ListBox entnommen und in eine Insert\_item Liste gespeichert. Je nach ausgewählten Wert in der jeweiligen ComboBox bzw. ListBox wird dann InsertInto oder InsertWithoutPK aufgerufen.

Delete nimmt als Parameter die strings table, pk\_column, pk\_sid und löscht die Spalte mit dem angegebenen Private Key aus der angegebenen Tabelle.

TransactionComplete setzt den bool newTransaction auf true.

# View

Im Ordner View befinden sich die Klassen, welche es dem Benutzer ermöglichen, Eingaben zu tätigen. Außerdem sind einige Hilfsklassen vorhanden. Diese Klassen sind:

* BaseMenu
* ComboItem
* CreateNewAnalysis
* DataView
* Insert\_item
* LoadForm
* Login
* SaveSchema
* SelectNavigationOperator
* SelectTable
* UserInput

Die oben aufgelisteten Klassen werden in den folgenden Kapiteln mit deren Feldern und Methoden genauer beschrieben.

# BaseMenu

Die Klasse BaseMenu öffnet vor dem eigenen Initialisieren ein neues Login Fenster. Wenn das Login fehlschlägt schließt sich das Programm automatisch. Erst wenn das Login Fenster erfolgreich geschlossen wurde öffnet sich das eigentliche BaseMenu.

Die Klasse BaseMenu hat eine Ansicht, auf der zwei Buttons und ein DataGridView vorhanden sind. Der Button mit dem Label „Create new schema“ ruft die Funktion CreateNewSchema auf. Der Button mit dem Label „Delete selected Schema“ ruft die Funktion DeleteSelectedSchema auf. Der DataGridView zeigt die verfügbaren Analyse-Schemen der Tabelle AGS\_ANALYSIS\_GRAPH\_SCHEMA an. Beim Klicken auf den Namen eines Analyse-Schemas öffnet sich das LoadForm.

Folgende Abbildung zeigt das BaseMenu ohne Beispieldaten:

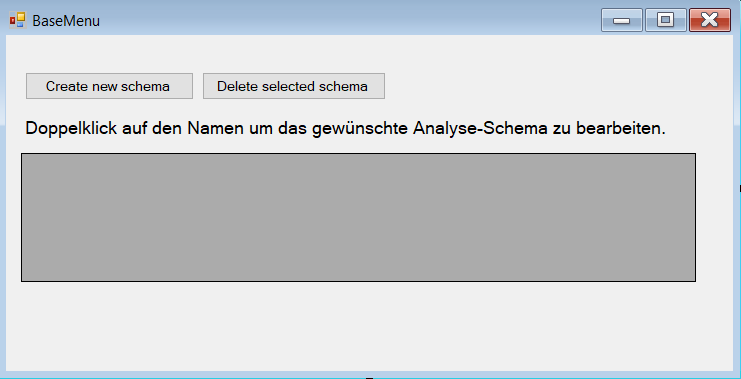


Abbildung 1: BaseMenu ohne Beispieldaten

Folgende Abbildung zeigt das BaseMenu mit Beispieldaten:

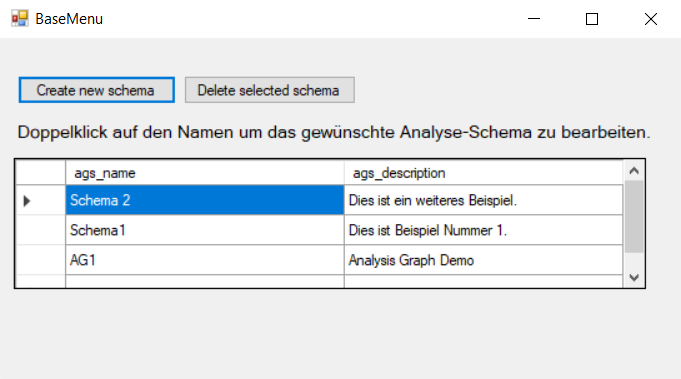


Abbildung 2: BaseMenu mit Beispieldaten

Die Klasse BaseMenu speichert folgende Daten:

* row speichert die Reihe, in welche geklickt wird, als int
* column speichert die Spalte, in welche geklickt wird, als int

CreateNewSchema erstellt ein neues CreateNewAnalysis Objekt und öffnet dessen Fenster.

DataGridViewCellContentClick

ValidateGraphSchema

ReloadGraphSchema ruft die Funktion ValidateGraphSchema auf.

DeleteSelectedSchema löscht das Schema, welches ausgewählt wurde.

DataGridViewCellMouseClick

# ComboItem

Die Klasse ComboItem speichert folgende Daten für die Verwendung in einer ComboBox:

* ID ist die ID des Items als int
* Text speichert den Text, der in einer ComboBox angezeigt wird, als string

Der Konstruktor nimmt die Parameter id, text und speichert diese in die dazugehörigen Felder.

SetComboboxContent erstellt aus den Parametern (combobox, table, column) eine DataTable und setzt die Daten der combobox dementsprechend.

Eine Überladung der Methode macht dasselbe mit zwei Spalten als Parameter, statt einer.

# CreateNewAnalysis

Die Klasse CreateNewAnalysis …

# Insert\_item

Die Klasse Insert\_item ermöglicht es einen Spaltennamen als string, und einen dazugehörigen Wert entweder als string oder int abzuspeichern.

Der Spaltenname wird unter column\_name als string gespeichert. Durch die Mitgabe eines string oder int im Konstruktor kann der jeweilige Wert entweder unter s\_value oder unter i\_value gespeichert werden. Wird ein String unter s\_value gespeichert, so bleibt in i\_value der Wert -404 gespeichert.

Durch die Methode IntValue wird über den i\_value ermittelt, ob ein int oder einen string Wert gespeichert wurde. Wenn der Wert -404 in i\_value gefunden wird ist es ein string, sonst ein int.

Mittels IsNull wird festgestellt ob das Insert\_item initialisiert wurde. Ist Is\_value null und i\_value -404 so ist es leer und gibt true zurück, ansonsten wird false zurückgegeben.

Durch GetColumnName wird der Spaltenname zurückgegeben.

Durch GetSValue wird der String Wert retourniert.

Durch GetIValue wird der Integer Wert retourniert.

# LoadForm

Die Klasse LoadForm …

# Login

Die Klasse Login ist für die Überprüfung der Login-Daten verantwortlich. Die Klasse speichert folgende Daten:

* loginSuccessful speichert ob das Login erfolgreich war als bool

Das Login hat mehrere Textfelder, in die die entsprechenden Login-Daten eingetragen werden können. Außerdem gibt es einen Button mit dem Label „Submit“, welcher die Methode Submit aufruft.

Folgende Abbildung zeigt das Login ohne Beispieldaten:

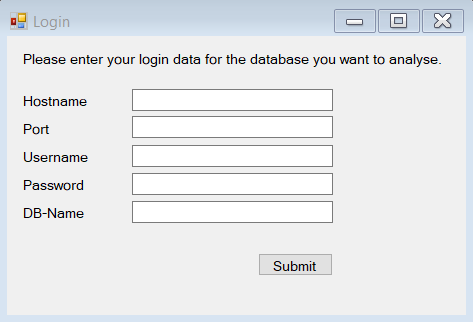


Abbildung 3: Login ohne Beispieldaten

Folgende Abbildung zeigt das Login mit Beispieldaten:

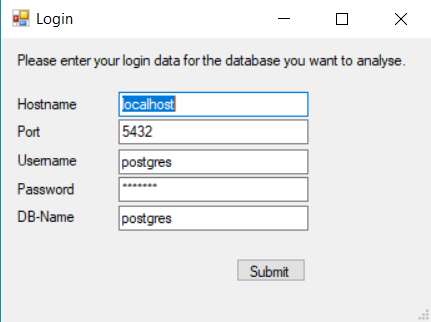


Abbildung 4: Login mit Beispieldaten

Der Konstruktor befüllt die Textfelder mit Beispieldaten.

Die Methode Submit nimmt die Daten aus den Textfeldern und versucht, eine Verbindung mit einer PostgreSQL Datenbank aufbauen. Ob der Aufbau erfolgreich war wird in loginSuccessful gespeichert. Das Fenster schließt sich danach.

# SaveSchema

Die Klasse SaveSchema …

# SelectNavigationOperator

Die Klasse SelectNavigationOperator macht es möglich, Operatoren auszuwählen und diese anzuwenden. Folgende Daten werden gespeichert:

* userInput speichert den UserInput, welcher als Parameter im Konstruktor übergeben wird als UserInput
* changed\_ComboBox speichert die ComboBox, welche verändert werden sollte als ComboBox
* changed\_TextBox speichert die TextBox, welche verändert werden sollte als TextBox
* changed\_ListBox speichert die ListBox, welche verändert werden sollte als ListBox
* agsNavstepSchema speichert den Namen des Operators als string
* selection speichert die Auswahl einer ComboBox als string
* schema speichert den Namen des Schemas als string
* dim\_sid speichert die DIM\_SID der ausgewählten ComboBox
* AGS\_NAVSTEP\_SCHEMA\_dictionary speichert zu jedem Namen der Operatoren den Namen der jeweiligen Tabelle als Dictionary<string, string>

Folgende Abbildung zeigt den SelectNavigationOperator ohne Beispieldaten:

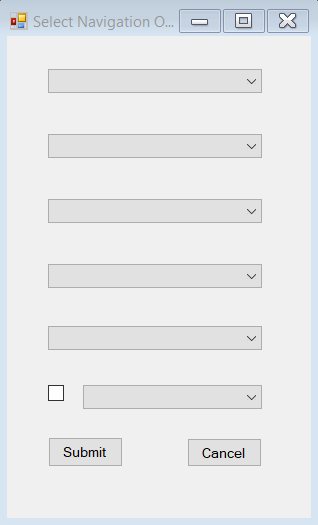


Abbildung 5: SelectNavigationOperator ohne Beispieldaten

Folgende Abbildung zeigt den SelectNavigationOperator mit Beispieldaten:

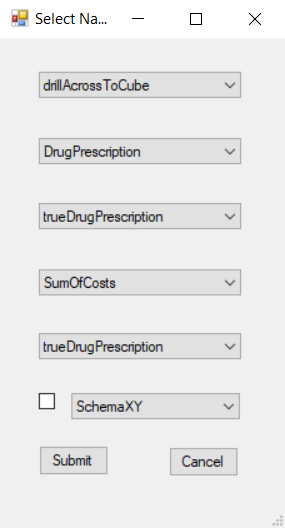


Abbildung 6: SelectNavigationOperator mit Beispieldaten

Der Konstruktor nimmt als Parameter einen UserInput ui. Dann ruft er die Methode FillDictionary auf. Er befüllt außerdem die ComboBox\_AgsNavstepSchema mit den Namen der Operatoren.

ComboBox\_AgsNavstepSchema\_SelectedIndexChanged speichert die Auswahl der ComboBox\_AgsNavstepSchema in agsNavstepSchema und füllt die ComboBox\_Selection entsprechend der Auswahl und setzt die entsprechende folgende ComboBox auf Visible.

ComboBox\_Selection\_SelectedIndexChanged speichert die Auswahl der ComboBox\_Selection in selection und füllt die ComboBox\_Selection2 entsprechend der Auswahl und setzt die entsprechende folgende ComboBox auf Visible.

ComboBox\_Selection2\_SelectedIndexChanged füllt die ComboBox\_Selection3 entsprechend der Auswahl und setzt die entsprechende folgende ComboBox auf Visible.

ComboBox\_Selection3\_SelectedIndexChanged füllt die ComboBox\_Selection4 entsprechend der Auswahl und setzt die entsprechende folgende ComboBox auf Visible.

ComboBox\_Selection4\_SelectedIndexChangedm setzt ComboBox\_AGS\_ANALYSIS\_SITUATION\_SCHEMA auf Visible.

ComboBox\_AGS\_ANALYSIS\_SITUATION\_SCHEMA\_SelectedIndexChanged speichert das ausgewählte Schema in schema ab.

Submit\_Click speichert die Tabelle mit dem entsprechenden Wert zum Operator in table. Dann wird eine Transaktion gestartet und speichert die ausgewählten Werte in die Tabelle table und in die Tabelle AGS\_NAVSTEP\_SCHEMA ab. Die Felder werden im UserInput deaktiviert um nicht mehr veränderbar zu sein. Dann wird das Fenster geschlossen.

Cancel\_Click schließt das Fenster, ohne zu speichern.

FillDictionary befüllt das AGS\_NAVSTEP\_SCHEMA\_dictionary mit allen möglichen Operatoren und deren dazugehörigen Tabellen.

# SelectTable

Die Klasse SelectTable …

# UserInput

Die Klasse UserInput …

# Program

Die Klasse Program besitzt die einzige Main Methode der Projektmappe, sprich, die Methode, welche das Programm an sich startet.

Die Main Methode ruft die Initialize Methode des DBContext auf. Dann wird die Run Methode der Application mit einem neuen BaseMenu als Parameter aufgerufen.

# Arbeitszeiten

Im folgenden Kapitel werden die Arbeitszeiten der einzelnen Personen mit kurzer Tätigkeitsbeschreibung tabellarisch dargestellt.



Abbildung : Arbeitszeiten Maria Limberger

# Abbildungsverzeichnis

[Abbildung 1: BaseMenu ohne Beispieldaten 9](#_Toc507436148)

[Abbildung 2: BaseMenu mit Beispieldaten 9](#_Toc507436149)

[Abbildung 3: Login ohne Beispieldaten 12](#_Toc507436150)

[Abbildung 4: Login mit Beispieldaten 12](#_Toc507436151)

[Abbildung 5: SelectNavigationOperator ohne Beispieldaten 14](#_Toc507436152)

[Abbildung 6: SelectNavigationOperator mit Beispieldaten 15](#_Toc507436153)

[Abbildung 7: Arbeitszeiten Maria Limberger 17](#_Toc507436154)