

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](#_Toc507756290)

[2. Beschreibung der Projektmappe 4](#_Toc507756291)

[2.1. Database 4](#_Toc507756292)

[2.1.1. DBContext 4](#_Toc507756293)

[2.1.2. DBService 5](#_Toc507756294)

[2.2. View 8](#_Toc507756295)

[2.2.1. BaseMenu 8](#_Toc507756296)

[2.2.2. ComboItem 10](#_Toc507756297)

[2.2.3. CreateNewAnalysis 11](#_Toc507756298)

[2.2.4. Insert\_item 12](#_Toc507756299)

[2.2.5. LoadForm 12](#_Toc507756300)

[2.2.6. Login 15](#_Toc507756301)

[2.2.7. SaveSchema 16](#_Toc507756302)

[2.2.8. SelectNavigationOperator 17](#_Toc507756303)

[2.2.9. SelectTable 20](#_Toc507756304)

[2.2.10. UserInput 20](#_Toc507756305)

[2.3. Program 25](#_Toc507756306)

[3. Arbeitszeiten 26](#_Toc507756307)

[4. Abbildungsverzeichnis 27](#_Toc507756308)

[5. Tabellenverzeichnis 28](#_Toc507756309)

# 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:

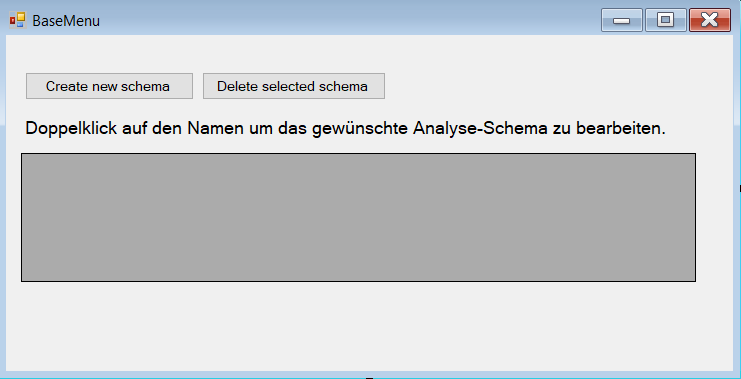


Abbildung 1: BaseMenu ohne Beispieldaten

Folgende Abbildung zeigt das BaseMenu mit Beispieldaten:

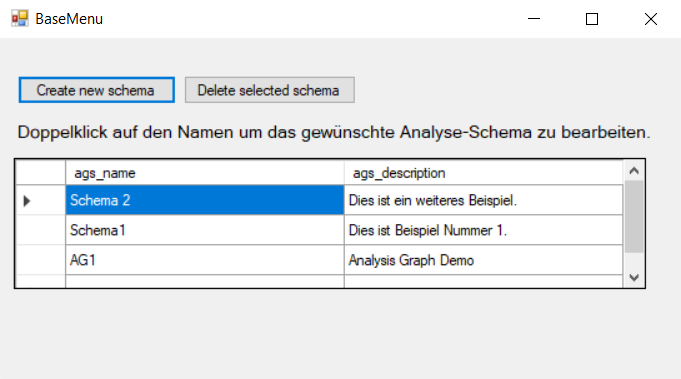


Abbildung 2: BaseMenu mit Beispieldaten

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 speichert in row und column die neue ausgewählte Spalte ab. Wenn der Spaltenindex 0 ist, dann wird das Graphen-Schema geladen und LoadForm aufgerufen.

ValidateGraphSchema erstellt eine Query die als Resultat alle aktuellen Analyse-Graph-Schemen (AGS\_ANALYSIS\_GRAPH\_SCHEMA) anzeigt.

ReloadGraphSchema ruft die Funktion ValidateGraphSchema auf und hält das DataGrid Feld aktuell.

DeleteSelectedSchema löscht das Schema, das in der abgespeicherten Zeile (row) und Spalte (column) ist.

DataGridViewCellMouseClick ruft ValidateGraphSchema auf und speichert den neuen Spalten- und Zeilenindex in row und column ab, wenn er größer als 0 ist.

# 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 ermöglicht es einen Namen und eine Beschreibung für ein neues Analyse-Graph-Schema einzugeben und anschließend in der Datenbank abzuspeichern.

Folgende Abbildung zeigt das CreateNewAnalysis Fenster ohne Beispieldaten:

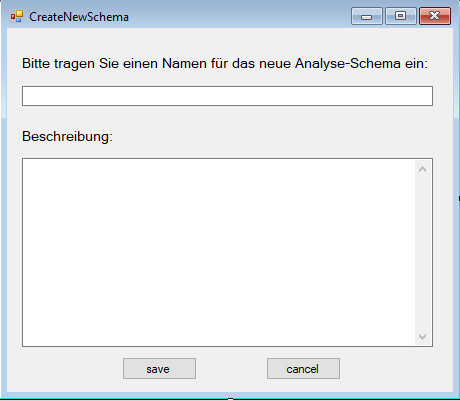


Abbildung 3: CreateNewSchema ohne Beispieldaten

Durch den saveButton werden die eingegebenen Daten aus beiden Textfeldern in die AGS\_ANALYSIS\_GRAPH\_SCHEMA Tabelle als neues Analyse-Graph-Schema gespeichert und das Fenster UserInput wird geöffnet.

Durch den cancelButton wird der Benutzer mittels MessageBox gefragt, ob er wirklich beenden will. Bei einem „Ja“ wird das CreateNewSchema Fenster geschlossen und BaseMenu erneut aufgerufen. Bei einem „Nein“ passiert nichts.

# 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 ermöglicht es eine gespeicherte Analyse-Situation auszuwählen, zu laden, zu löschen, zu bearbeiten oder eine neue zu erstellen.

Die folgende Abbildung zeigt das LoadForm Fenster ohne Beispieldaten:

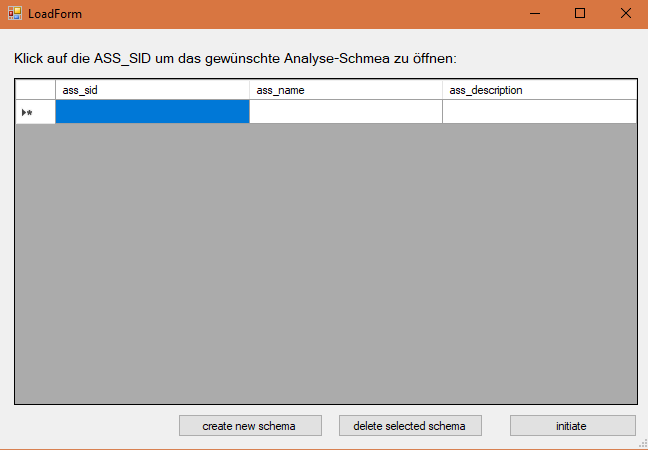


Abbildung 4: LoadForm ohne Beispieldaten

Die folgende Abbildung zeigt das LoadForm Fenster mit Beispieldaten:

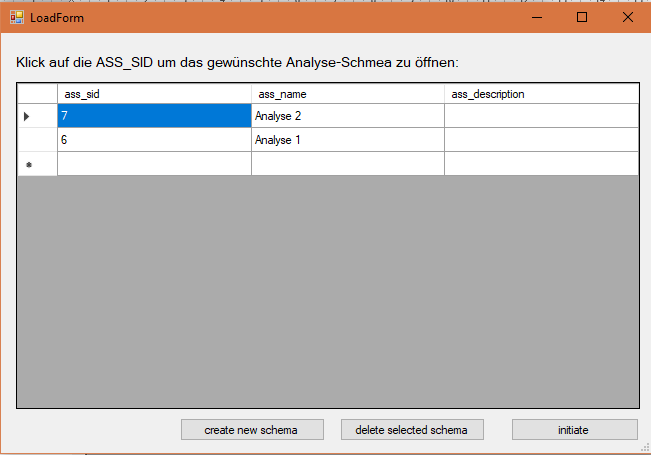


Abbildung 5: LoadForm mit Beispieldaten

LoadForm speichert außerdem folgende Daten für die Verwendung in UserInput und für ein DataGridView:

- ags\_sid speichert den zuvor in BaseMenu erstellten oder geladenen Analyse-Graph-Schema Schlüssel (ags\_sid).

- row speichert die Reihe, in welche geklickt wird, als int

- column speichert die Spalte, in welche geklickt wird, als int

LoadForm wird eine ags\_sid übergeben die diese in ags\_sid speichert.

Durch validateGraphSchema wird eine Query erstellt, die die aktuellen Analyse-Schema die zur zuvor (BaseMenu) ausgewählten Analyse-Graf-Schema passen im DataGridView anzeigt.

Durch dataGridView1\_CellContentClick wird der neue Wert der Reihe und Spalte in row und column gespeichert, wenn dieser jeweils größer als 0 ist. Ist der Spaltenindex genau 0 wird das Analyse-Schema geladen indem UserInput aufgerufen wird.

Durch LoadForm\_Click wird validateGraphSchema aufgerufen und der DataGridView aktualisiert.

Durch create\_new\_schema\_Click wird UserInput aufgerufen.

Durch delete\_selected\_schema wird der User noch einmal durch eine MessageBox gefragt, ob er das Schema wirklich löschen will. Bei ja wird das ausgewählte Schema, das in row und column sich befindet, gelöscht.

Durch dataGridView1\_Cell\_MouseClick wird die aktuelle ausgewählte Zelle im DataGridView in column und row gespeichert.

Durch initiateButton\_Click wird von der ausgewählten Zeile der ass\_sid ermittelt und UserInput übergeben. Somit wird das Schema instanziiert.

# 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:

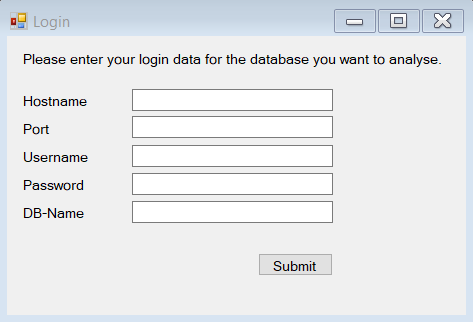


Abbildung 6: Login ohne Beispieldaten

Folgende Abbildung zeigt das Login mit Beispieldaten:

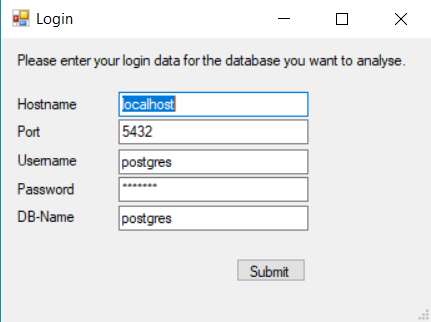


Abbildung 7: Login mit Beispieldaten

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 ist für die Speicherung des Analyse-Schema Namens und dessen Beschreibung zuständig. Weiters speichert die Klasse folgende Daten:

* ui speichert den UserInput, durch den die Klasse aufgerufen wurde.

Folgende Abbildung zeigt das SaveSchema Fenster ohne Beispieldaten:

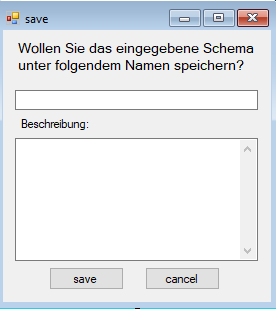


Abbildung 8: SaveSchema ohne Beispieldaten

Durch save\_button\_Click wird abgefragt, ob überhaupt ein Name eingegeben wurde, wenn nein dann wird User erneut dazu aufgefordert. Falls das Schema geladen wurde und somit das Feld (overrideSchema) in UserInput true ist wird der User gefragt, ob er entweder das Schema überspeichern will oder ob er es als neues Schema abspeichern möchte. Danach wird das Analyse-Schema unter dem eingegebenen Namen und der Beschreibung gespeichert.

Durch cancel\_button\_Click wird das SaveSchema Fenster geschlossen.

# 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:

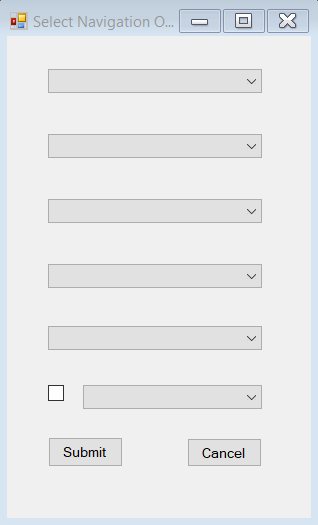


Abbildung 9: SelectNavigationOperator ohne Beispieldaten

Folgende Abbildung zeigt den SelectNavigationOperator mit Beispieldaten:

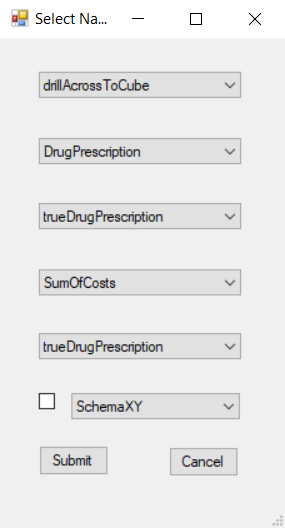


Abbildung 10: SelectNavigationOperator mit Beispieldaten

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 ermöglicht das Eingeben, Auswählen und Speichern von Daten. Dazu verwendet es mehrere Attribute:

* loaded\_ags\_sid speichert die ags\_sid des Analyse-Graf-Schemas als int
* loaded\_ass\_sid speichert die ass\_sid des Analyse-Schemas als int
* overrideSchema ist ein boolean der bei true in der Klasse SaveSchema Auswirkungen hat
* name speichert den Standardnamen „userinput“ als String und wird durch SaveSchema anders belegt
* description speichert die Beschreibung des Analyse-Schemas als String und ist anfänglich leer.
* dim\_doctor, dim\_insurant, dim\_drug, dim\_medservice, dim\_hospital, dim\_time sind boolean die festlegen ob die Dimension gerade möglich ist (true) oder nicht (false)
* newForm ist ein boolean der bei einem neuen Schema true ist und bei einem alten false
* bmsr\_var, filter\_var sind boolean die bei true die jeweilige ListBox freigeben und bei false diese sperren (ausgrauen)
* Für jede Dimension (Dimension Qualification) gibt es CheckBox mit einem dazugehörigen boolean mit der Endung \_var, z.B.: time\_DL\_var.
* selectTable (Philip)

Der Konstruktor UserInput erstellt, wenn newForm true ist, eine neue Analyse und lädt alle möglichen Cubes in die CubeComboBox oder ruft load auf.

comboBoxCube\_SelectedIndexChanged ruft folgende Methoden auf, wenn der User eine Auswahl beim CubeComboBox Feld trifft: disable\_dimensions, bmsr\_Instantiate, measures\_Instantiate, filter\_Instantiate, dimension\_enable\_disable.

disable\_dimesions schaltet alle Felder, die für die Dimension Qualification benötig werden, aus.

disable\_fields schaltet alle Felder, die durch die Checkbox variabel wurden, aus.

dimension\_enable\_disable überprüft mittels einer Query, welche Dimensionen beim ausgewählten Cube erlaubt sind und aktiviert diese. Dabei wird die Dice Level (DL) ComboBox der jeweiligen Dimensionen befüllt.

bsmr\_Instantiate lädt die auswählbaren BMSR-Filter in die ListBox LDW\_BMSR.

measure\_Instantiate lädt die auswählbaren Measures in die ListBox LDW\_MEASURES.

filter\_Instantiate lädt die auswählbaren Filter in die ListBox LDW\_FILTER.

Die folgenden Methoden sind zur Befüllung der Dimension Qualifications:

fillComboboxDimension führt eine Query durch, die als Ergebnis den Dice Level (DN) der mitgegebenen Dimensions ComboBox füllt.

fillListBoxDimension führt eine Query durch, die als Ergebnis die Slice Condition (SC) der mitgegebenen Dimensions ListBox füllt.

fillGL führt eine Query durch, die den Granularity Level (GL) der jeweiligen mitgegebenen Dimensions ComboBox füllt.

Tabelle 1: \_SelectedIndexChanged

|  |  |
| --- | --- |
| CDW\_TIME\_SelectedIndexChanged  CDW\_INSURANT\_SelectedIndexChanged  CDW\_MEDSERVICE\_SelectedIndexChanged  CDW\_DRUG\_SelectedIndexChanged  CDW\_DOCTOR\_SelectedIndexChanged  CDW\_HOSPITAL\_SelectedIndexChanged | Diese Methoden befüllen und halten die jeweilige Slice Condition (SC) und den jeweiligen Granularity Level (GL) der Dimension aktuell, indem sie fillGL und fillListBoxDimension aufrufen. |

TDW\_TIME\_KeyPress stellt sicher, dass bei der Zeitdimension im Textfeld des Dice Nodes (DN) nur Zahlen eingebbar sind.

deselect ermöglich durch das Pressen der Leertaste das aktuell selektierte zu deselektieren.

button\_cancel\_Click fragt den User mittels MessageBox, ob er wirklich das Fenster schließen will. Bei einem ja wird das Fenster geschlossen und das BaseMenu aufgerufen.

button\_select\_navigation\_operator\_Click ermöglicht das Aufrufen des SelectNavigationOperator Fensters.

button\_save\_Click ruft SaveSchema auf um das Schema zu benennen und zu speichern.

load setzt overrideSchema auf true und lädt das Schema aus Abfragen und befüllt die betreffenden ComboBoxen, ListBoxen, Textfelder, Checkboxen und wählt die gespeicherten Felder aus.

insert bereitet eine Transaktion vor, wo alle ausgewählten Felder in die dafür vorgesehenen Datenbanken abgespeichert werden. Dafür benützt die Funktion viele Methoden der DBService Klasse. Bei einem Error wird die genaue Error-Nachricht in der Konsole ausgegeben.

overrideDB ruft deleteSchema auf und anschließend insert.

deleteSchema löscht das Schema mittels Delete aus der Klasse DBService.

Tabelle 2: \_CheckedChanged

|  |  |
| --- | --- |
| time\_DL\_CheckedChanged  insurant\_DL\_CheckedChanged  meds\_DL\_CheckedChanged  hospital\_DL\_CheckedChanged  doctor\_DL\_CheckedChanged  drug\_DL\_CheckedChanged | Diese Methoden sperren den jeweiligen GL (Granularity Level) der Dimension, wenn die Checkbox des jeweiligen DL (Dimension Level) angekreuzt wurde bzw. gibt sie diesen wieder frei. |
| bmsr\_variable\_CheckedChanged  filter\_variable\_CheckedChanged  time\_DN\_CheckedChanged  time\_SC\_CheckedChanged  time\_GL\_CheckedChanged  insurant\_DN\_CheckedChanged  insurant\_SC\_CheckedChanged  insurant\_GL\_CheckedChanged  meds\_DN\_CheckedChanged  meds\_SC\_CheckedChanged  meds\_GL\_CheckedChanged  drug\_DN\_CheckedChanged  drug\_SC\_CheckedChanged  drug\_GL\_CheckedChanged  doctor\_DN\_CheckedChanged  doctor\_SC\_CheckedChanged  doctor\_GL\_CheckedChanged  hospital\_DN\_CheckedChanged  hospital\_SC\_CheckedChanged  hospital\_GL\_CheckedChanged | Diese Methoden setzten die jeweiligen boolean Felder mit der Endung „\_var“ auf angekreuzt (true) oder nicht angekreuzt (false). |

getAssSid gibt den ass\_sid als int zurück.

button1\_Click (Philip)

DisableVars ?

# 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.

Tabelle 3: Arbeitszeiten Maria Limberger



# Abbildungsverzeichnis

[Abbildung 1: BaseMenu ohne Beispieldaten 9](#_Toc507756063)

[Abbildung 2: BaseMenu mit Beispieldaten 9](#_Toc507756064)

[Abbildung 3: CreateNewSchema ohne Beispieldaten 11](#_Toc507756065)

[Abbildung 4: LoadForm ohne Beispieldaten 13](#_Toc507756066)

[Abbildung 5: LoadForm mit Beispieldaten 13](#_Toc507756067)

[Abbildung 6: Login ohne Beispieldaten 15](#_Toc507756068)

[Abbildung 7: Login mit Beispieldaten 16](#_Toc507756069)

[Abbildung 8: SaveSchema ohne Beispieldaten 17](#_Toc507756070)

[Abbildung 9: SelectNavigationOperator ohne Beispieldaten 18](#_Toc507756071)

[Abbildung 10: SelectNavigationOperator mit Beispieldaten 19](#_Toc507756072)

# Tabellenverzeichnis

[Tabelle 1: \_SelectedIndexChanged 22](#_Toc507756073)

[Tabelle 2: \_CheckedChanged 23](#_Toc507756074)

[Tabelle 3: Arbeitszeiten Maria Limberger 26](#_Toc507756075)