**INCLUDE zcm\_inc\_generate\_s0\_top.**  
Top-Include.  
Enthält eine Variable lv\_stgnr, die vom Typ eine ID eines Studiengangs (SC) ist.

**INCLUDE zcm\_inc\_generate\_s0\_screen.**  
Enthält den Aufbau des Selection-Screens

**INCLUDE zcm\_inc\_generate\_s0\_f01.**  
Enthält die Logik zur Ausführung der Aktionen

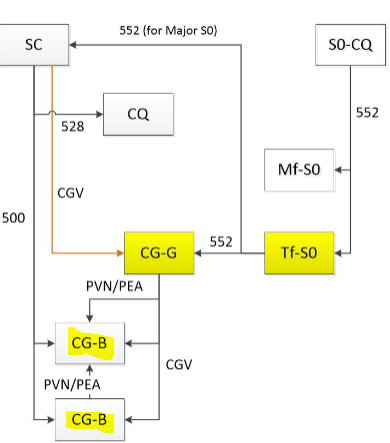
Hier wird ein LOOP AT gebraucht, da bei Select-Options eine Mehrfachselektion ausgewählt werden kann.

**Klasse z\_cm\_cl\_s0\_creation\_helper**

Methode: create\_s0s\_for\_sc

Aufruf der Methode get\_relevant\_cg\_for\_sc

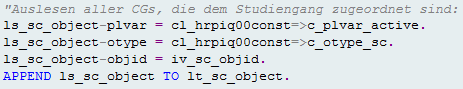
Hier wird ermittelt, für welche CGs die Studienangebote angelegt werden sollen

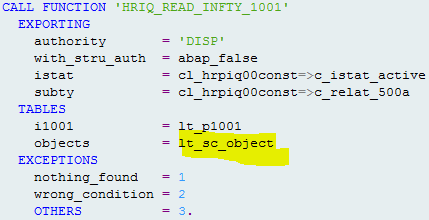


Die Felder der Struktur „ls\_sc\_object“ werden gefüllt mit Konstanten Werten.

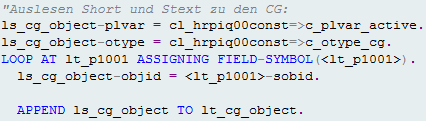
Objekttyp = SC (Studiengang).

Da die Methode „HRIQ\_READ\_INFTY\_1001“ als Übergabeparameter eine Tabelle benötigt, wird die Struktur zur internen Tabelle „lt\_sc\_object“ angehängt.



Nun werden alle 1001-Infotypen, deren Subtyp „A500“ ist, gelesen.  
Hier werden alle Verknüpfungen des Studiengangs (SC) abgefragt.  
  


Anschließend werden aus der internen Tabelle lt\_p1001, in der nun alle Einträge enthalten sind, die die richtige Verknüpfungsart haben, alle Einträge, deren SCLAS nicht ‚CG‘ ist, gelöscht (es sollen ja CGs, die mit diesem Studiengang verknüpft sind, in der internen Tabelle sein).  
  


Nun wird eine Struktur für den Infotyp 1000 angelegt.  
Dazu werden alle gefundenen CG-Verknüpfungen in einer Loop-Schleife durchlaufen und die Objektids zur Struktur zugeordnet und diese wird anschließend an eine Tabelle (lt\_cg\_object) angehängt.  


Der Funktionsaufruf „HRIQ\_READ\_INFTY\_1000“ wird benötigt, da im 1000er Infotyp der SHORT-hinterlegt ist, der zwingend benötigt wird.  
  
Nachdem man eine Tabelle für den Infotyp 1000 hat, braucht man noch eine für den Infotyp 1733 (Enthält unter anderem die Modulgruppenart). Diese wird wieder ausgehend von den CG-Objekten befüllt.

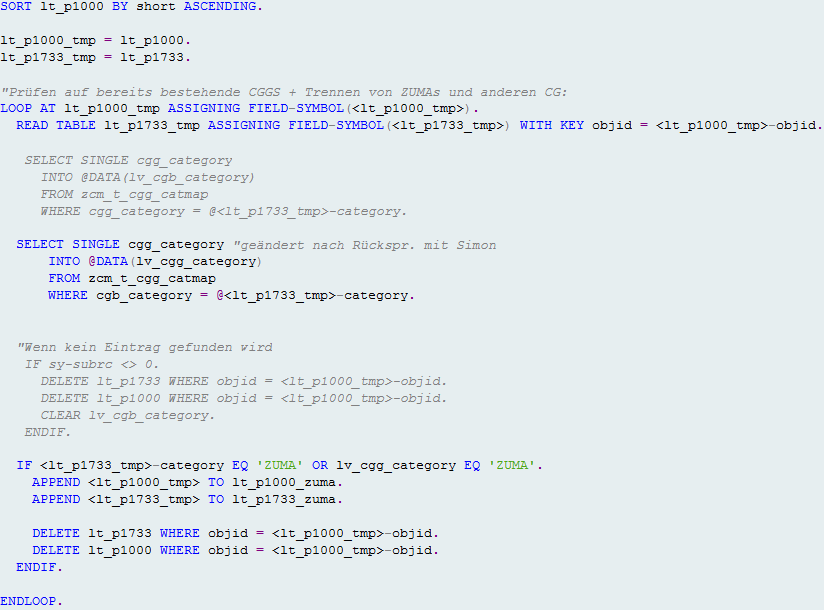
**D.h. Durch die Selektion auf die Verknüpfungen bekommt man alle CGs, die dem SC zugeordnet sind. Anschließend muss man nur noch mit den CGs weiterarbeiten. Bei den hier gefundenen CGs handelt es sich um die CG-Bs (siehe Grafik, S. 1)**

Nun hat man folgende Tabellen:

Lt\_p1000 (Enthält die Daten der CGs)  
LT-p1001 (Alle Verknüpfungen mit Typ CG)  
LT\_p1733 (Enthält die Modulgruppenarten der CGs)

Im nächsten Schritt wird die Tabelle p1000 auf Basis des Shorts aufsteigend sortiert.

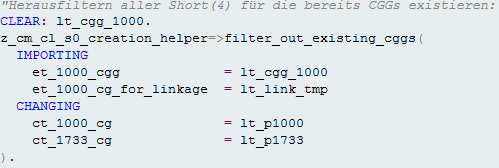
Dann werden die Werte von lt\_p1000 und lt\_p1733 jeweils einer tmp-Tabelle zugewiesen.



Jetzt wird die tmp-Tabelle des Infotyps 1000 geloopt.  
In der Select-Anweisung wird nach der Zuteilung zwischen der „normalen“ Kategorie der vorhandenen Infotypen und der „CGG-Kategorie“ herausgesucht.  
Ist eine der Kategorien „ZUMA“ werden die beiden Werte jeweils an eine spezielle Zuma-Tabelle angehängt und aus der „normalen“ Tabelle gelöscht, damit diese nicht doppelt verarbeitet werden.

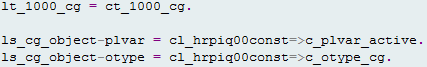
Als nächstes wird die Methode „filter\_out\_existing\_cggs“ aufgerufen.  
Hier werden die Tabellen lt\_p1000 und lt\_p1733 als changing-Parameter übergeben (sie werden in der Methode direkt bearbeitet und so zurückgegeben).

Nach Ausführen dieser Methode erhält man eine Liste für alle Verknüpfungen, die noch angelegt werden müssen (lt\_link\_tmp) und eine Liste, in der alle bereits existierenden CGGs enthalten sind (lt\_cgg\_1000).

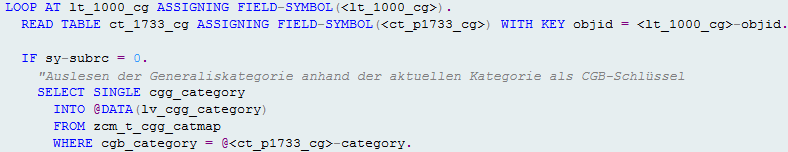


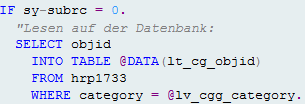
Hier gibt es auch wieder eine interne Tabelle lt\_1000\_cg, die die übergebenen CGs aus der oberen Methode enthält (Durch Zuweisung über den Importparameter ct\_1000\_cg).

Anschließend werden die Werte für ein CG-Objekt wieder mit Konstanten vorbelegt.

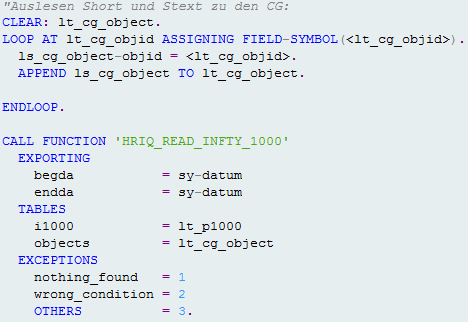


Nun wird wieder über die lt\_1000\_cg geloopt.  
Die Tabelle ct\_1733\_cg wird anhand der Objektid des aktuellen Wertes von lt\_1000\_cg gelesen.

Ist die Objektid vorhanden, wird die Generalisationskategorie aus der Mapping-Tabelle gelesen.  


Wenn ein Eintrag in der Mapping-Tabelle gefunden wurde, werden alle Objektids der Einträge in der Tabelle HRP1733 gelesen, die die Kategorie der zuvor ermittelten Generalisationskategorie haben.  
  


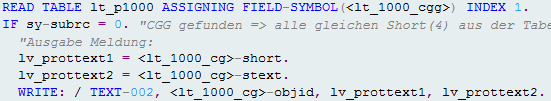
Nun wird über die soeben ermittelten Objektids geloopt und dem ls\_cg\_object zugeordnet. Die anderen Werte wurden bereits am Beginn dieser Methode initialisiert. Jedes Objekt wird wiederum an eine Tabelle angehängt, die benötigt wird, um die zugehörigen Infotypen zu lesen.



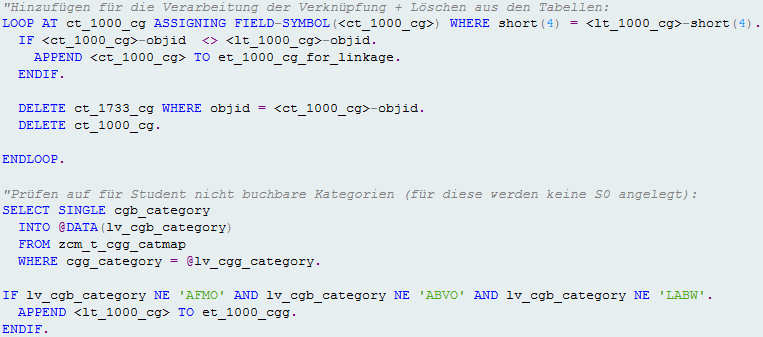
Nun sind in der Tabelle lt\_p1000 die Infotypen zu den gefundenen Einträgen, die die gesuchte Generalisationskategorie haben.

Hier werden alle Einträge gelöscht, deren Short-Nummer (die 4-stellige am Anfang) unterschiedlich zur aktuellen Zeile der Loop-Schleife ist.



Danach wird die Tabelle am Index 1 gelesen (damit wird überprüft, ob überhaupt ein Eintrag vorhanden ist).  
Wird nichts gefunden, wird der Schleifendurchaluf beendet.  


Wird etwas gefunden, wird die übergebene Tabelle für jeden Short(4), der mit dem Short(4) des aktuellen Schleifendurchlaufs geloopt.

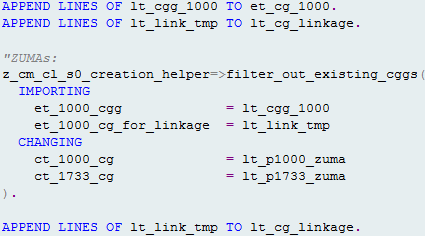


Dann wird die Objektid verglichen. Gibt es zu einem Short(4) mehrere Einträge, wird dieser zur Tabelle „et\_1000\_cg\_for\_linkage“ hinzugefügt.

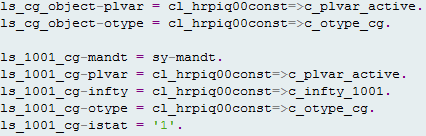
DELETE ct\_1000\_cg bewirkt, dass die Zeile, die aktuell im Loop durchlaufen wird, gelöscht wird.

Wenn die gefundene Kategorie für die Studenten buchbar ist (also alle außer die abgefragten), werden diese der Exporttabelle „et\_1000\_cgg“ hinzugefügt. D.h. für diese CGs muss noch ein CGG angelegt werden (oder?)

Nach dieser Methode werden die beiden Tabellen lt\_cgg\_1000 und lt\_link\_tmp and die Tabellen et\_cg\_1000 bzw. lt\_cg\_linkage angehängt und die Methode erneut ausgeführt, diesmal jedoch mit den Zuma-Tabellen.



Methode PROCESS\_1001\_FOR\_EXISTING\_CGG

Hier wird das ls\_cg\_object wieder mit konstanten Werten vorbelegt.  
Außerdem wird eine Struktur für den Infotyp 1001 bereits vorbelegt.  




Anschließend wird die Methode Process\_cgs aufgerufen, die unter anderem die Anlage der CGGs beinhaltet.