

# SuperX-Entwicklerhandbuch



[www.MemText.de](http://www.MemText.de)

- Daniel Quathamer  
[danielq@memtext.de](mailto:danielq@memtext.de)
- Meikel Bisping  
[mbisping@memtext.de](mailto:mbisping@memtext.de)

<http://www.superx-projekt.de>

Version 3.5  
Stand 28.5.2007

Download als [PDF-Datei](#)



<b>1</b>	<b>Historie .....</b>	<b>6</b>
1.1	Neuigkeiten in SuperX 3.5 .....	6
<b>2</b>	<b>Erzeugung und Änderung von Masken .....</b>	<b>6</b>
2.1	Ein Tutorial .....	7
2.1.1	Ausgangspunkt .....	8
2.1.1.1	Das Beispiel .....	8
2.1.1.2	Hintergründe .....	9
2.1.1.2.1	Die Felddefinitionen .....	9
2.1.1.2.2	Speichern der Felddefinition: die Tabelle felderinfo .....	11
2.1.1.2.3	Änderung einer Felddefinition .....	14
2.1.1.3	Maskendefinition .....	15
2.1.1.3.1	Abfragen in Maskendefinitionen .....	17
2.1.1.3.2	Änderung einer Abfrage .....	19
2.1.2	Fazit .....	21
2.2	Erweiterte Maskenprogrammierung: Freemarker Templates .....	21
2.2.1	Klassische Verarbeitung .....	22
2.2.2	FreeMarker Transformation .....	22
2.2.2.1	Übersicht .....	22
2.2.2.2	Programmieren mit FreeMarker .....	23
2.2.2.2.1	Zugriff auf Java-Objekte im Datenmodell .....	24
2.2.2.2.2	if-Abfragen .....	24
2.2.2.2.3	Variablen .....	24
2.2.2.2.4	has_content .....	27
2.2.2.2.5	ForEach .....	27
2.2.2.2.6	For ...Next ...-Schleifen: List .....	27
2.2.2.2.7	Makros und Funktion .....	28
2.2.2.3	Special tricks .....	28
2.2.2.4	SQL-Lingua Franca .....	28
2.2.2.5	Allgemeine FM-Makros/Funktionen .....	30
2.2.2.6	Spezielle Möglichkeiten bei Sicht-Feldern .....	30
2.2.2.6.1	allNeededKeys – für temporäre Datentabellen .....	30
2.2.2.6.2	keysToRoot – für Verteilschritte .....	31
2.2.2.6.3	elements– für Schleife über ausgewählte Knoten .....	31
2.2.2.6.4	Zugriff auf einzelne Knoten im Baum .....	32
2.2.2.7	Grundgerüst für neue Abfragen .....	33
2.2.3	Datenbankunabhängigkeit .....	34
2.2.4	Zugriff auf Konstantentabelle und Hochschulinfo .....	35
2.2.5	Sx_repository .....	35
2.2.6	Abfragen mit variabler Spaltenzahl .....	36
2.2.7	Tooleinsatz .....	39
2.2.7.1	Jedit .....	39
2.2.7.1.1	FreeMarker-Syntax Highlighting .....	40
2.2.7.1.2	Folding .....	40
2.2.7.2	sx_masken_sql_update.x .....	40
2.3	Abfragenentwurf mit SuperX-Sichten .....	41
2.3.1	Einträge verstecken oder nicht-selektierbar machen .....	43
2.3.2	User-/Gruppenrechte .....	44

2.3.3	Benutzung der Sichten in Masken .....	44
2.3.4	Alt. Hierarchien aus CoB .....	45
2.4	Spezielle Details .....	45
2.4.1	Felder auf der Maske verstecken .....	45
2.5	Abfragemakros (einschl. Schleifen u. Grafiken).....	46
2.5.1	Makros und Sichten.....	47
2.5.2	Makro-Schachtelung.....	47
2.5.3	Schleifenfunktion .....	48
2.5.3.1	Grundlagen.....	48
2.5.3.2	Schleife über ein anderes Makro.....	48
2.5.4	Spezielle Auswahlwerte hinterlegen.....	49
2.5.5	Zukünftig: Feldnamen-Synonyme .....	50
2.5.6	Aktionen (Grafikerzeugung).....	51
2.5.6.1	Grundlagen.....	51
2.5.6.2	Grafikerstellung .....	51
2.5.6.2.1	Grundlagen.....	51
2.5.6.2.2	MoreAttribs.....	53
2.5.6.2.3	Säulendiagramme.....	53
2.5.6.2.4	Balkendiagramme .....	53
2.5.6.2.5	Tortendiagramme .....	54
2.5.6.3	spezielle Stylesheets benutzen .....	54
2.6	Dokumentation von Abfragen .....	55
2.6.1	Glossare .....	55
2.6.1.1	Allgemeine Schlüsselwörter .....	55
2.6.1.2	Der Spezialfall Maskenfelder.....	56
2.6.1.3	Änderung von Glossaren im XML-Frontend.....	56
2.6.1.3.1	Maskenerläuterung.....	56
2.6.1.3.2	Feldbeschriftungen ändern .....	61
2.6.2	Erzeugung der SuperX-Hilfe im Javahelp-Format .....	62
2.7	Werkzeuge zur Masken-Entwicklung .....	63
2.7.1	Shell-Scripte .....	63
2.7.1.1	Masken-Verwaltung.....	63
2.7.1.1.1	Eine Maske suchen .....	63
2.7.1.1.2	Eine Maske sichern und entladen.....	63
2.7.1.1.3	Eine Maske neu einfügen.....	64
2.7.1.1.4	Eine Maske löschen .....	65
2.7.1.2	Änderungen an einer Maske vornehmen.....	65
2.7.2	Das Access-Frontend.....	65
2.7.3	Maskenverwaltung mit MS Access .....	66
2.7.4	Weitere Tools .....	67
2.7.4.1	SQLWorkbench .....	68
2.7.5	Entwicklungsservlet für SuperX-Abfragen .....	69
2.7.5.1	Aufrufseite des Entwicklungsservlets .....	69
2.7.5.2	Funktionalität des Entwicklungsservlets.....	70
2.7.5.3	Berechtigung für das Entwicklungsservlet.....	72
2.7.6	Masken für das XML-Frontend vorbereiten.....	73
2.7.6.1	Erzeugen eines Stylesheets .....	73
2.7.6.2	Zuordnung einer Maske zu einem Stylesheet .....	74

2.7.6.3	Anpassung an Lesegeräte.....	75
2.7.6.4	Erweiterungen des XML-Frontends.....	77
2.7.6.4.1	Navigationsspalten im XML-Frontend .....	77
2.7.6.4.2	Hierarchieebenen in Ergebnisspalten.....	79
2.7.6.4.3	PDF-Export.....	81
2.7.6.4.4	Excelexport.....	82

# 1 Historie

## 1.1 Neuigkeiten in SuperX 3.5

Es gibt ein einige Neuerungen in SuperX3.5:

- Anlegen von [Abfragen mit variabler Spaltenzahl](#) (S. 36)
- Anlegen von [Hierarchieebenen](#) (S. 79) in Spalten (einschließlich Aufklappmöglichkeit)
- Möglichkeit zum Anlegen [versteckter oder nicht selektierbarer Einträge](#) (S. 43) in Sichten
- Spezielles [Entwicklungsservlet](#) (S. 69)

## 2 Erzeugung und Änderung von Masken

Die Abfragemasken liefern die Daten aus den Basissystemen an das SuperX-Frontend aus. Einige Abfragen zur Administration sind im Kernmodul enthalten, die Abfragen zu den Basissystemen sind in den jeweiligen Modulen enthalten. Die Abfragen in der Administration erlauben es, neue Masken anzulegen, zu kopieren und zu löschen.



Um den Austausch von Abfragen innerhalb der Hochschulen zu erleichtern ("Abfragen-Pooling" über die SuperX-Website), sollten die Masken immer im Nummernkreis xxxx0000 bis xxxx9990 liegen, wobei xxxx der von der HIS verwandten Hochschulnummer entspricht. Die Zehnerschritte ergeben sich daraus, dass die dazwischen liegenden Nummern für die Maskenfelder (Tabelle `felderinfo`) reserviert sind<sup>1</sup>.

Im Folgenden finden Sie allgemeine Hinweise für die Verwaltung der Masken.

Die Masken lassen sich browserbasiert, über UNIX-Shellscripte, und über Access administrieren.

Weitergehende Möglichkeiten bietet aber das XML-Frontend (Möglichkeit der Editierung von großen `text`-Feldern bei Postgres als Datenbanksystem). Nach der Anmeldung haben Administratoren das Recht, Masken zu löschen, zu kopieren und erzeugen. Die einzelnen Felder der Masken lassen sich direkt in der Datenbank oder z.B. mit [MS Access](#) (S. 66) verändern. Im Applet sind nur grundlegende Verwaltungsoperationen möglich. Sie sind als Ersatz für die [UNIX-Scripte](#) gedacht.

Folgende "Abfragen" zur Maskenverwaltung gibt es im Sachgebiet Administration: Darunter im Ast "Felder" gibt es noch folgende Abfragen: Darüberhinaus gibt es (nur unter Postgres) die Masken zur Pflege von Masken bzw. Feldern	<ul style="list-style-type: none"> <li>• Maske kopieren</li> <li>• Maske löschen</li>   <li>• Feld kopieren</li> <li>• Feld löschen</li> <li>• Maske suchen</li> <li>• Feld suchen</li> </ul>
--	---

**Maske kopieren.** Wie im [UNIX - Script](#) (S. 64) wird eine Maske in eine neue Maske kopiert, und alle zugehörigen Tabellen werden aktualisiert. Zusätzlich wird auch der Eintrag im Themenbaum gemacht. Bei der Nummer der Maske (tid) sollten Sie das Nummernschema von SuperX einhalten, um in Zukunft [Abfragen-Pooling](#) (S. 6) zu ermöglichen.

**Maske löschen.** Wie im [UNIX-Script](#) (S. 65) werden Masken aus allen dazugehörigen Tabellen entfernt. Zusätzlich wird auch der Eintrag im Themenbaum gelöscht. Zur Sicherheit muss die Nummer der Maske manuell eingegeben werden.

**Maske suchen.** Sie können Masken suchen und im XML-Frontend komfortabel editieren. Schränken Sie Ihre Auswahl auf ein Sachgebiet ein, und drücken Sie "Abschicken". Sie erhalten eine Liste mit "Treffern", und rechts befinden sich jeweils Buttons zum ansehen bzw. editieren einer Maske. Die Maske läuft nur unter Postgres, weil Informix kein direktes Bearbeiten von Blob-Feldern mit sql unterstützt.

**Feld suchen.** Sie können analog zu "Maske suchen" auch Felder suchen und bearbeiten.

Die Abfragen sind selbsterklärend; das Erzeugen neuer Masken, Löschen vorhandener Masken und Kopieren vorhandener Masken ist nur für Userkennungen möglich, die in der Tabelle `userinfo` im Feld `administration` den Wert 1 haben. Natürlich sollten die Abfragen sehr vorsichtig benutzt werden, sie sind die einzigen Abfragen in SuperX, die tatsächlich Änderungen an der Datenbank vornehmen können.

## 2.1 Ein Tutorial

Im Folgenden wollen wir zeigen, wie Abfragemasken in SuperX arbeiten und wie man die Ergebnisdarstellung von Abfragen verändern kann. Wir zeigen dies am Beispiel der Abfrage **Studierende (Zeitreihe)**.

Erforderliche Kenntnisse:

- SQL und Datenbankbedienung
- Grundkenntnisse zu SuperX

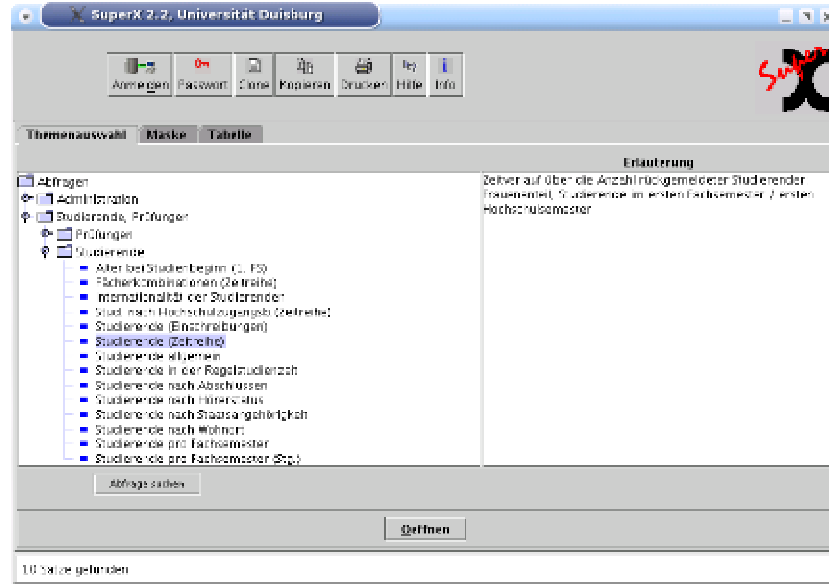
## 2.1.1 Ausgangspunkt

### 2.1.1.1 Das Beispiel

Der Ausgangspunkt ist ein Beispiel aus dem SOS-Modul.

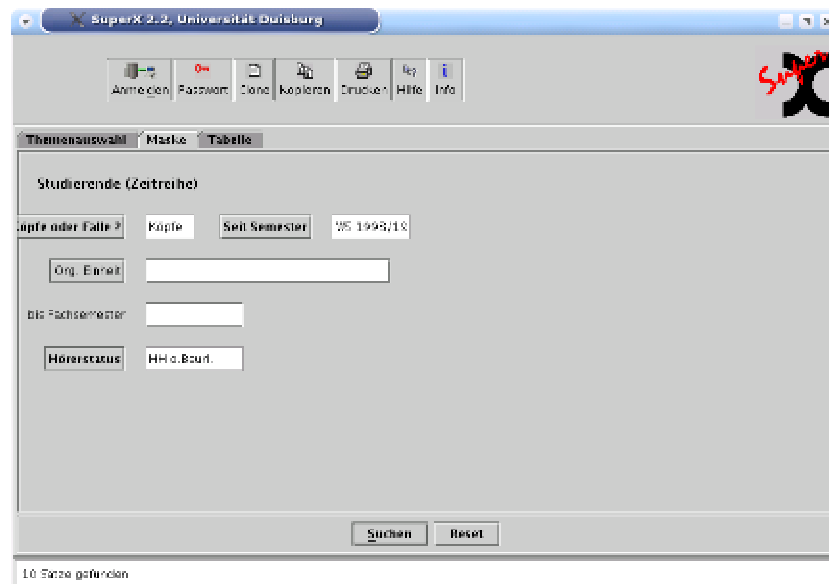
(Klicken Sie jeweils auf die Grafiken, um sie zu vergrößern).

Wir wählen aus dem Themenbaum im Bereich Studierende die Abfrage "Studierende (Zeitreihe)".



Die Abfrage liefert eine Statistik über Studierende im Laufe von mehreren Semestern, je nach 1. Fachsemester, 1. Hochschulsesemester und Geschlecht. Die folgende Abbildung zeigt die Maske:

Die Abbildung zeigt die Auswahlfelder der Maske. Wir wählen für den Zeitraum im Feld "Seit Semester" das WS 1998/1999.



Wenn wir "Suchen" drücken, erscheint folgende Ergebnisdarstellung:



Die Tabelle zeigt in der ersten Spalte die Semester, dann die Gesamtzahl der Studierenden und die Studierenden im 1. Fachsemester.

SuperX 2.2.2, Universität Duisburg

Themenauswahl Maske Tabelle

Studierende (Zeitreihe)

Parameter:  
 Käpfe oder Fächer ? = Käpfe Seit Semester = FS 1990/1998 Orig. Einheit = keine Auswahl - Stand 30.01.2005 Hörerstatus = III a,IIII;IIII=unpers;

Stand: 19.03.2005

Semester	Gesamtzahl	1. FS gesamt	1. FS in %	1. HS gesamt	1. HS in %	dar. Frauen	Frauen in %	1. FS Frauen	1. FS Frauen in %	1. HS Frauen	1. HS Frauen in %
SS 2003	13655	158	1,15	63	0,46	5487	40,18	85	53,46	78	44,44
WS 2002	15049	3106	20,64	2427	16,13	8051	40,21	1381	44,70	1108	45,57
SS 2002	13661	535	3,92	348	2,55	5371	39,32	263	49,16	136	39,08
WS 2001...	14324	2722	19,00	2230	15,57	5681	39,69	1221	49,81	978	45,86
SS 2001	12862	426	3,31	210	1,64	5010	38,98	203	47,65	97	46,19
WS 2000...	13737	2312	16,84	1787	13,00	5338	38,85	885	42,59	763	42,46
SS 2000	11055	448	4,04	216	1,95	5000	45,23	235	51,04	110	31,82
WS 1999...	13902	2143	15,41	1583	11,39	5353	38,50	823	43,07	694	43,84
SS 1999	13166	485	3,69	161	1,20	5085	37,79	252	51,90	86	33,42
WS 1998...	12882	1692	13,14	1469	11,42	4835	37,54	849	42,62	633	45,09

10 Einträge gefunden

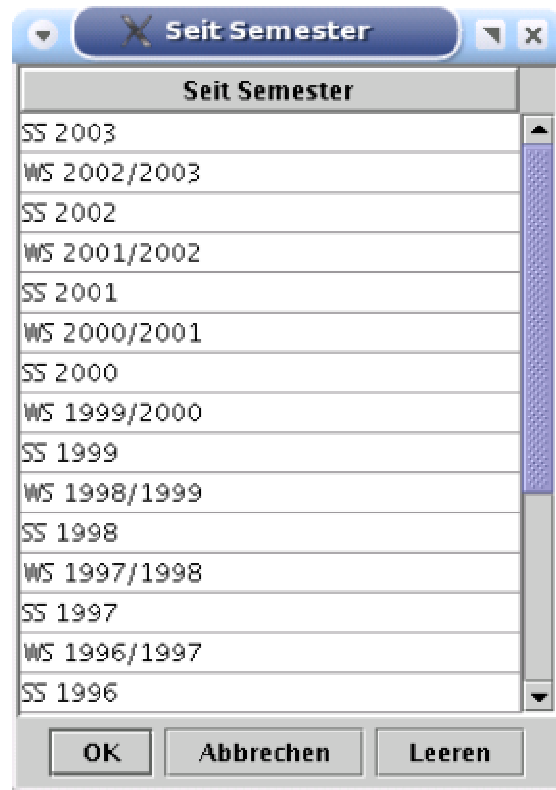
### 2.1.1.2 Hintergründe

Wie werden nun Felder in den Auswahlmasken gefüllt, und wie werden die Ergebnisse in SuperX ermittelt?

#### 2.1.1.2.1 Die Felddefinitionen

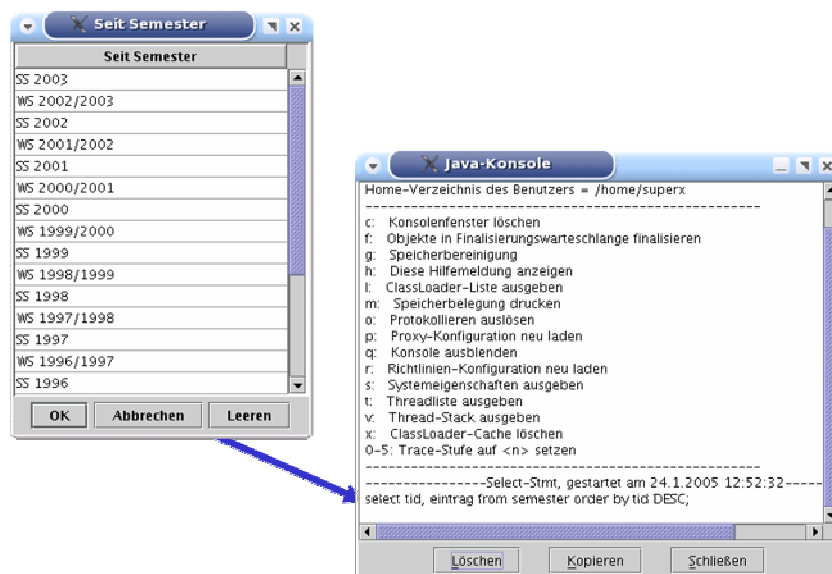
Gehen wir kurz zurück zur Auswahlmaske. Jedes Feld der Maske, z.B. "Seit Semester", ist ein Datensatz in der Tabelle `felderinfo`. Dort finden Sie Angaben zum Namen, Inhalt und Layout des Feldes. Gehen wir zunächst zum Inhalt des Feldes: Die Liste der Semester.

Beim Klick auf das Feld Semester erhalten wir eine Reihe von Semestern zur Auswahl. Die Liste ist absteigend sortiert.



Um den Inhalt des Feldes zu erläutern, wollen wir kurz auf eine nützliche Funktion bei der Abfragentwicklung hinweisen, die Java-Konsole. Wenn Sie die Java-Konsole in der Systemsteuerung aktiviert haben, dann können Sie im Browser die Konsole anzeigen lassen. Im Mozilla z.B. gehen Sie in das Menü "Werkzeuge" -> "Web-Entwicklung" und dort auf "Java Konsole". Im Internet Explorer machen Sie einen Doppelklick auf das Apfelmännchen bzw. eine Kaffeetasse (Java-Symbol je nach Java-Version) unten rechts in der Shortcut-Leiste des Betriebssystems.

Beim Klick auf den Button "Semester" sehen wir in der Konsole folgenden SQL-Befehl unten rechts.



Die SQL-Anweisung liefert aus der Tabelle `semester` die Felder `tid` ("Tupelidentifizier") und `eintrag` (der Volltext des Semesters). Der Schlüssel des Feldes `tid` ist unsichtbar, sorgt aber dafür, dass die Sortierung richtig erfolgt.

Hier sehen Sie einen Screenshot der Tabelle `semester` (Auszug) direkt in der Datenbank. Die Nummerierung ist fünfstellig und besteht aus Jahr (vier Stellen) und 1 für Sommer- und 2 für Wintersemester.

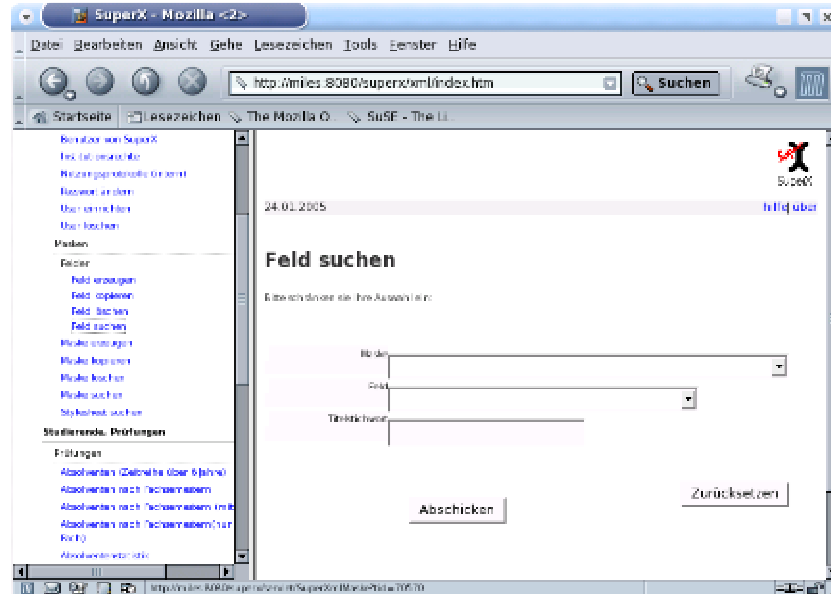
tid	eintrag	
19921	SS 1992	...
19922	WS 1992/1993	...
19931	SS 1993	...
19932	WS 1993/1994	...
19941	SS 1994	...
19942	WS 1994/1995	...
19951	SS 1995	...
19952	WS 1995/1996	...
19961	SS 1996	...
19962	WS 1996/1997	...
19971	SS 1997	...
19972	WS 1997/1998	...
19981	SS 1998	...
19982	WS 1998/1999	...
19991	SS 1999	...
19992	WS 1999/2000	...
20001	SS 2000	...
20002	WS 2000/2001	...
20011	SS 2001	...
20012	WS 2001/2002	...
20021	SS 2002	...
20022	WS 2002/2003	...
20031	SS 2003	...

#### 2.1.1.2.2 Speichern der Felddefinition: die Tabelle `felderinfo`

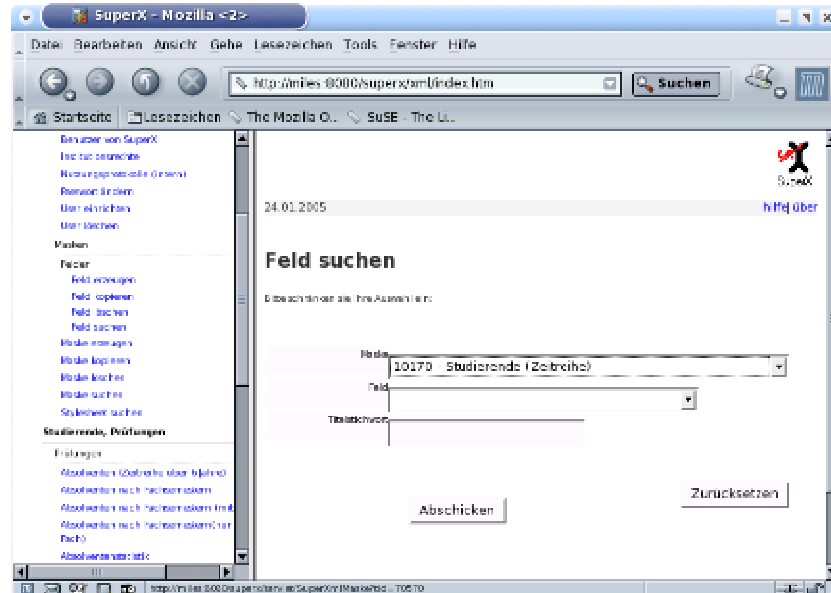
Wo wird nun in SuperX die Felddefinition gespeichert? Viele Skripte in SuperX werden selbst in Datenbanktabellen abgelegt, die Tabelle `felderinfo` enthält die relevanten Angaben für die Felder.

Um dies zu sehen, öffnen wir ein Formular im XML-Frontend, dort befinden sich Bearbeitungsformulare für Felder und Masken.

Im Themenbaum des XML-Frontends finden wir den Menüpunkt "Feld suchen". Rechts erscheint ein leeres Formular.

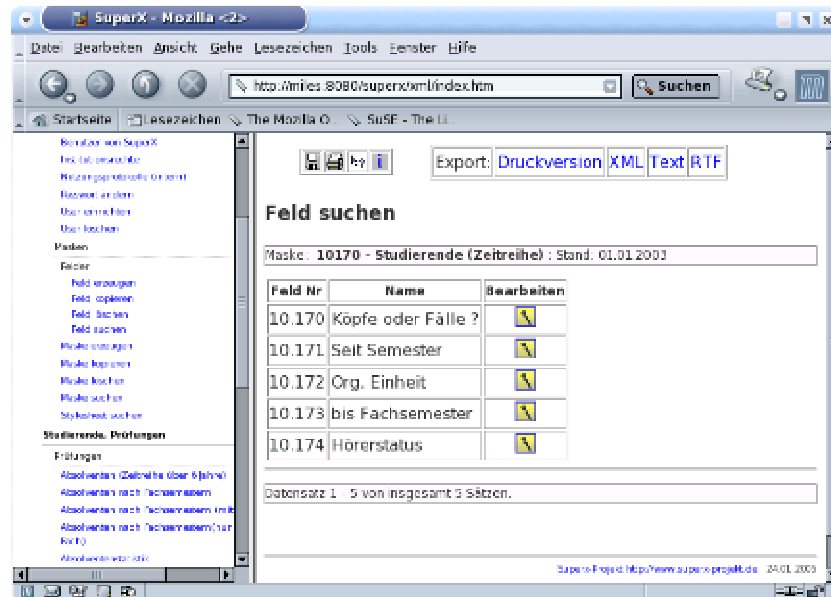


In dem Formular wählen wir die Abfrage Studierende Zeitreihe aus. Zusätzlich sehen wir auch die Nummer der Maske (10170), das ist bei der Maskenbearbeitung ganz nützlich.



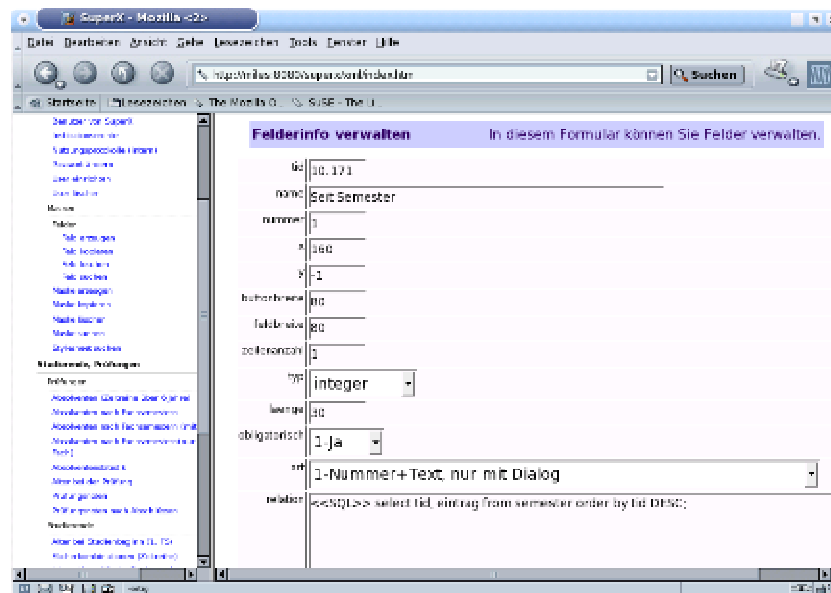
Wenn wir hier Abschicken drücken, erscheint folgendes Bild:

Die Maske enthält fünf Felder; wir sehen die Nummer des Feldes und den Namen. Rechts daneben befindet sich ein Knopf zum Bearbeiten des Feldes.



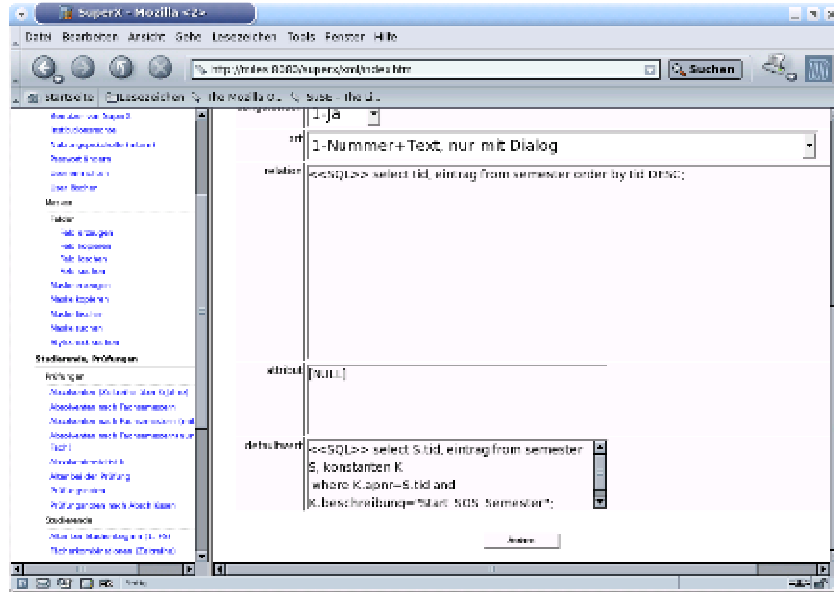
Ein kleiner Hinweis an dieser Stelle: Die Felder werden in der Tabelle `masken_felder_bez` der Maske Nr. 10170 zugeordnet. Wir zählen also bei Feldnummern in Einer-Schritten von der Maskennummer aus hoch. Aus diesem Grunde wählen wir bei Maskennummern Intervalle von mindestens 10, die nächste Maske wäre also mit 10180 nummeriert.

Wir wählen nun das Feld "Seit Semester", und gelangen in ein Bearbeitungsformular der Tabelle `felderinfo`. Wir sehen Name, Nummer, Position auf der Maske, Breite und Typ des Feldes (ganzzahlig). Das Feld ist obligatorisch, und von der Art Nr. 1 (Nummer + Text, mit Dialog).



Ganz unten sehen Sie das Feld "relation", in dem nach einem Steuerungszeichen `<<SQL>>` der SQL-Befehl steht, den wir vorhin in der Java-Konsole gesehen haben.

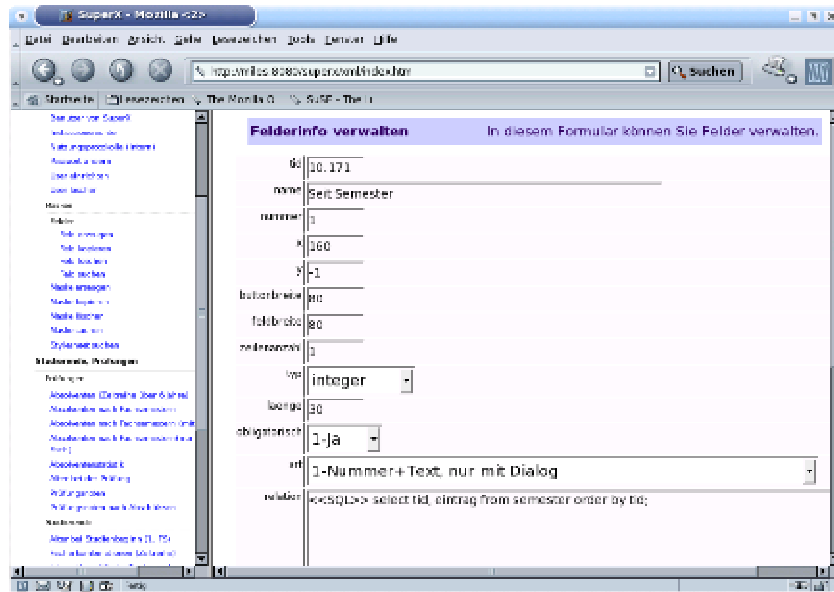
Der Vollständigkeit halber zeigen wir hier den Rest der Tabelle. Unten ist noch der Defaultwert für das Feld angegeben, ebenfalls ein SQL-Ausdruck.



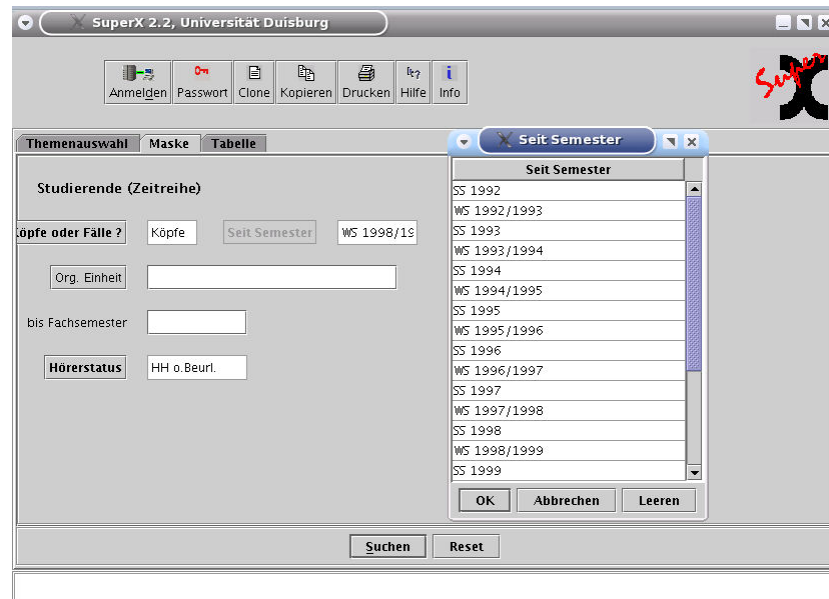
SuperX liest also aus der Datenbank die Skripte für eine Maske bzw. für ein Feld aus einer Tabelle, und führt Sie dann in der Datenbank aus.

### 2.1.1.2.3 Änderung einer Felddefinition

Wir können nun sparsenshalber das Feld ändern, um die Sortierung anzupassen. Wir löschen im field Relation das Wort "DESC", und schicken die Änderung ab.



Nun müssen wir im Applet die Maske einmal neu in der Themenauswahl öffnen, dann wird das neue Script aus der Datenbank geladen. Wenn wir dann in der Maske auf "Seit Semester" klicken, erscheinen die Semester in aufsteigender Reihenfolge.



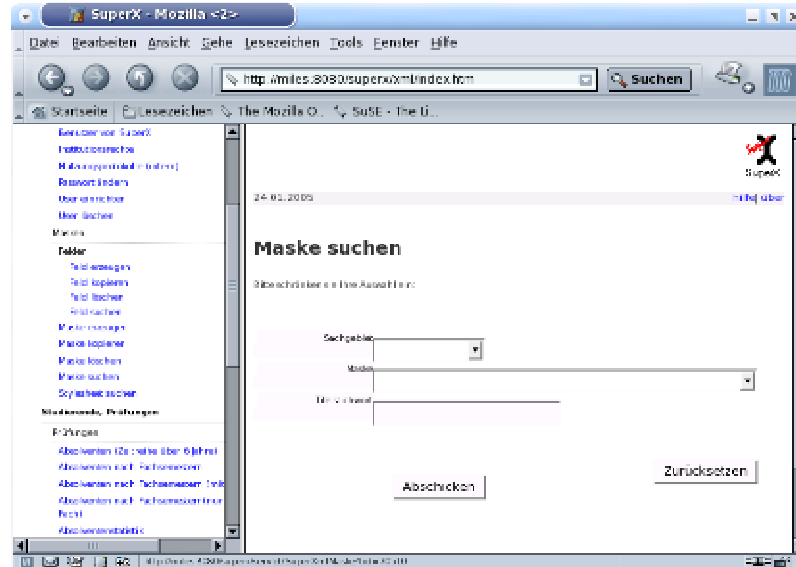
Auf diese Art und Weise können wir alle Maskenfelder bearbeiten. Die restlichen Attribute in der Tabelle `felderinfo` sind im **Administrationshandbuch Kernmodul** erläutert.

### 2.1.1.3 Maskendefinition

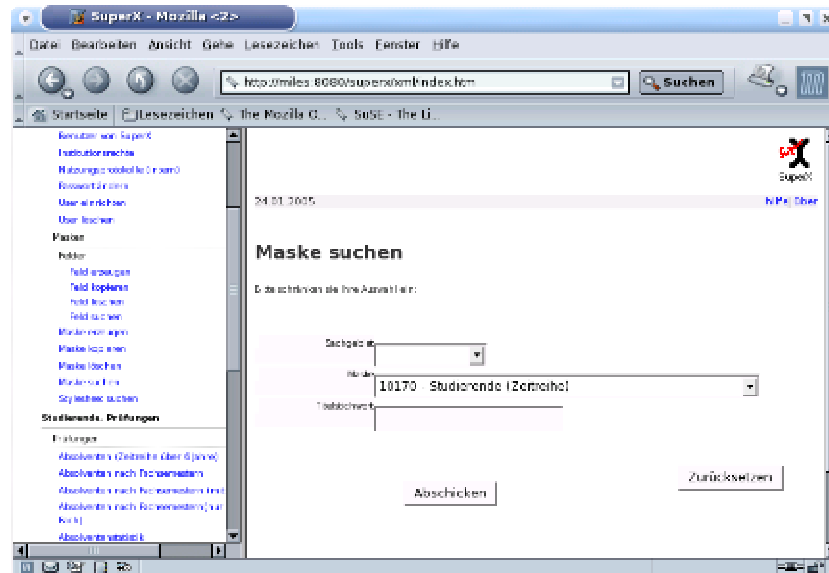
So weit so gut, wir können nun also Felder ändern. Wie können wir nun die Ergebnistabellen bearbeiten?

SuperX arbeitet hier ebenfalls mit SQL-Skripten, die als Felder in einer Tabelle gespeichert sind. Die Tabelle lautet `maskeninfo`. Wir können uns diese Tabelle ebenfalls im XML-Frontend anschauen:

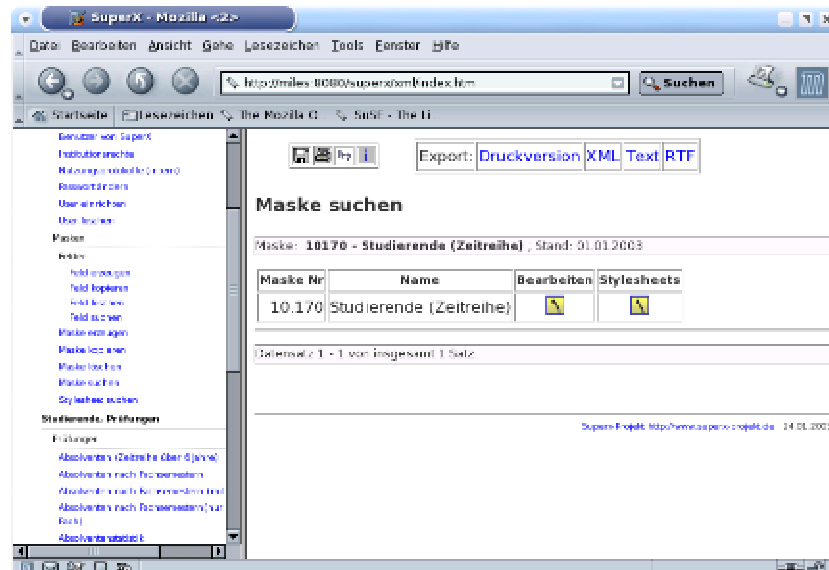
Im Themenbaum wählen wir Maske suchen. Es erscheint rechts eine Auswahlmaske ohne Vorbelegung.



Dort wählen wir wieder die Abfrage **Studierende (Zeitreihe)** aus.

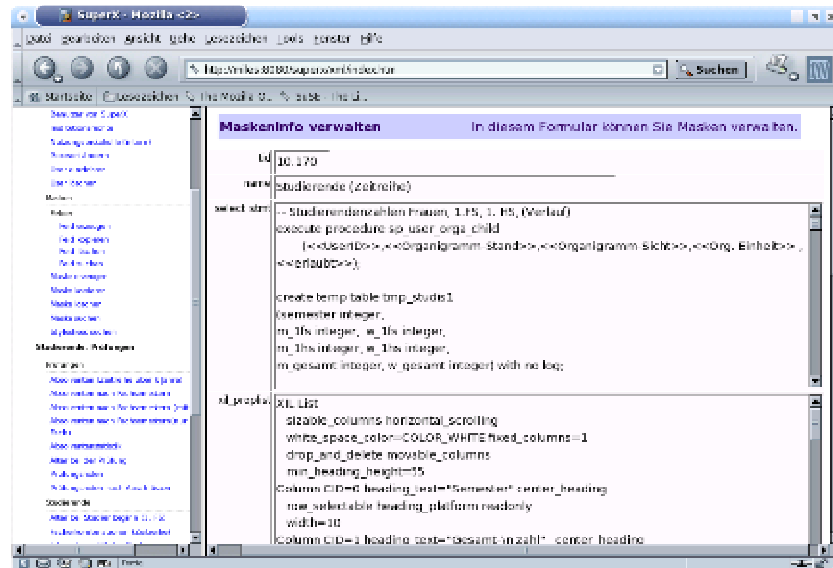


Als Ergebnis sehen wir unsere Maske sowie zwei Bearbeitungsbuttons. Wir wählen den ersten Button, **Bearbeiten**.





Wir gelangen in das Bearbeitungsformular der Maske. Neben der Nummer der Maske sehen wir den Namen und die Felder `select_stmt` und `xil_proplist`. Das Feld `select_stmt` enthält das SQL-Script, und `xil_proplist` die Ergebnisdarstellung.



### 2.1.1.3.1 Abfragen in Maskendefinitionen

Das Script in `select_stmt` ist relativ lang, wir wollen es daher nur auszugsweise kommentieren. Allgemein formuliert arbeiten wir so:

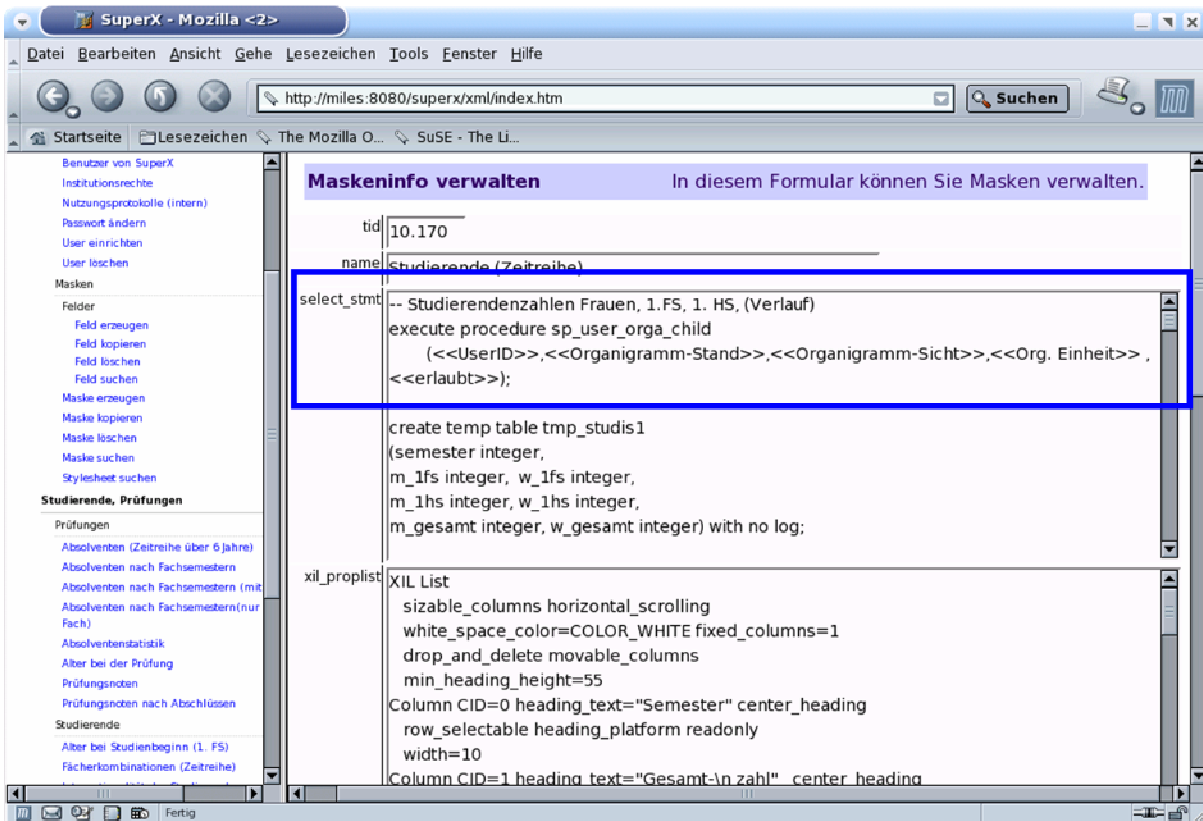
1. Zunächst werden die Eingaben in den Auswahlfeldern ausgewertet und eine Ergebnismenge ermittelt, meist in Form einer temporären Tabelle.
2. Diese Tabelle wird mit den Hilfstabellen in der Datenbank gejoined, und es wird eine Ergebnistabelle berechnet. Ggf. werden noch Summen oder Prozente berechnet, meist benötigen wir dazu weitere temporäre Tabellen.
3. Der letzte `select` im Feld `select_stmt` enthält die Ergebnistabelle, die das Applet empfängt. Mit Hilfe der `xil_proplist` werden die Spaltenüberschriften- und Breiten gesetzt, und das Ergebnis wird angezeigt.
4. Direkt danach wird die letzte temporäre Tabelle gedroppt, und die Datenbankverbindung wird an das SuperX-Servlet zurückgegeben.

Am Anfang eines SQL-Scriptes werden die Auswahlfelder ausgewertet, die der Anwender angeklickt

hat, bevor er **Suchen** gedrückt hat. So wird z.B. das Feld

Org. Einheit

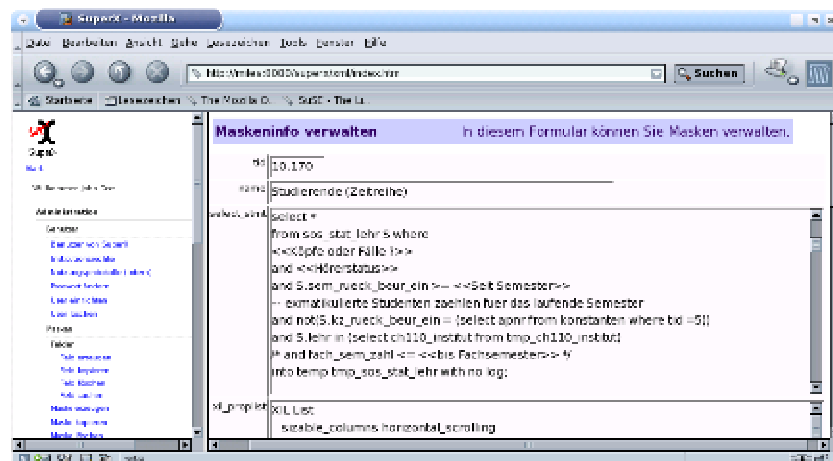
wie folgt interpretiert:



Die Prozedur "execute procedure..." steht am Anfang von fast jeder SuperX-Abfrage und ermittelt eine temporäre Tabelle tmp\_ch110\_institut, die die ausgewählten Lehreinheitsnummern enthält<sup>ii</sup>. Konkret wird der Passus "<<Org. Einheit>>" durch den Schlüssel ersetzt, der in der Maske ausgewählt wurde. Es handelt sich also bei SuperX-Abfragen um dynamisches SQL.

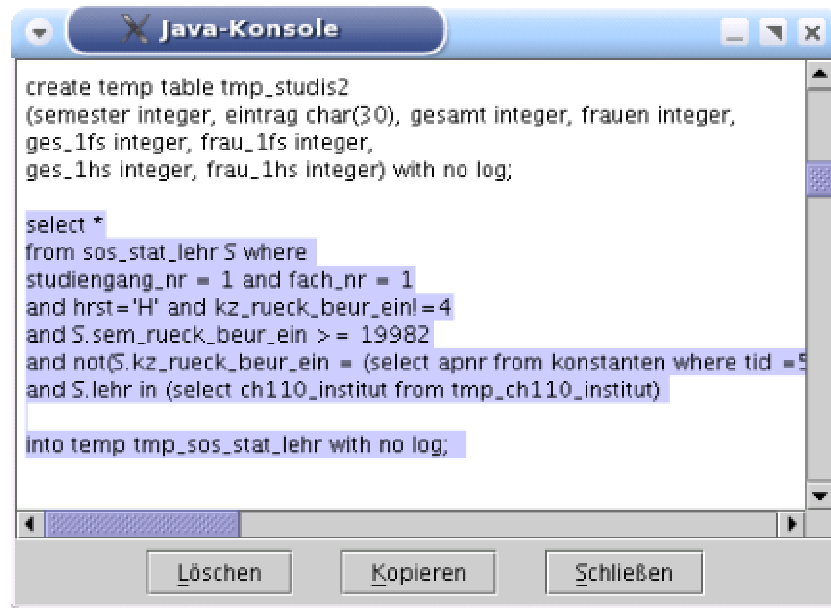
Die Lehreinheiten wiederum werden dann mit der Hilfstabelle sos\_stat\_lehr gejoined, die eine Statistik von allen Lehreinheiten und Semestern enthält<sup>iii</sup>. Die Einschränkungen durch die Maskenfelder Köpfe oder Fälle, Seit Semester und Hörerstatus sieht in SQL wie folgt aus:

Für "Köpfe oder Fälle" etc. finden wir hier nur Platzhalter. Alle relevanten Sätze werden in die temporäre Tabelle tmp\_sos\_stat\_lehr selektiert (Syntax von Informix, bei Postgres sieht das etwas anders aus).



Wenn das Script abläuft, werden die Platzhalter ersetzt. In der Java-Konsole sieht das so aus:

Statt "<<Köpfe oder Fälle>>" finden wir den SQL-Ausdruck `studiengang_nr=1 and fach_nr=1`, der **Hörerstatus** ist auf "H" codiert, und die **Beurlaubten** (Status 4) werden ausgefiltert. Das **Semester** muss  $\geq 19982$  sein, die **Exmatrikulierten** (Status 5) werden ebenfalls ausgefiltert, und die relevanten **Lehrheiten** werden ausgewählt.



```

create temp table tmp_studis2
(semester integer, eintrag char(30), gesamt integer, frauen integer,
ges_1fs integer, frau_1fs integer,
ges_1hs integer, frau_1hs integer) with no log;

select *
from sos_stat_lehr S where
studiengang_nr = 1 and fach_nr = 1
and hrst='H' and kz_rueck_beur_einl=4
and S.sem_rueck_beur_ein >= 19982
and not(S.kz_rueck_beur_ein = (select apnr from konstanten where tid = 5)
and S.lehr in (select ch110_institut from tmp_ch110_institut)

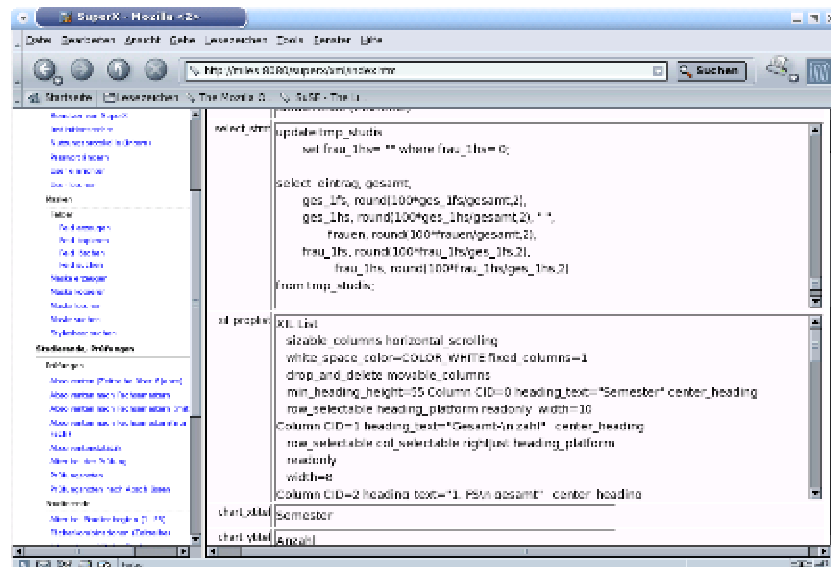
into temp tmp_sos_stat_lehr with no log;

```

Der letzte Select enthält die Ergebnistabelle: "select eintrag...".

Das Feld `eintrag` enthält den Volltext des Semesters, `gesamt` die Gesamtsumme, `ges_1fs` die Gesamtsumme der Studierenden im 1.FS etc.

In der XIL-Proplist sehen Sie schon die Überschriften in der Ergebnistabelle.



```

update tmp_studis2
set frau_1hs=0 where frau_1hs=0;

select eintrag, gesamt,
ges_1fs, round(100*ges_1fs/gesamt,2),
frauen, round(100*frauen/gesamt,2),
frau_1fs, round(100*frau_1fs/ges_1fs,2),
frau_1hs, round(100*frau_1hs/ges_1hs,2)
from tmp_studis2;

xil_proplist:
XIL_List
sizeable, columns horizontal scrolling
white space, color=COLOR, width fixed, columns=1
drop and delete movable, columns
min_heading_height=15 Column CID=0 heading_text="Semester" center_heading
row_selectable heading_platform readonly width=30
Column CID=1 heading_text="Gesamt" center_heading
row_selectable col_selectable right justify heading_platform
readonly width=8
Column CID=2 heading_text="1. FS absamt" center_heading
width=8
char_xml
Semester
char_xml
Gesamt

```

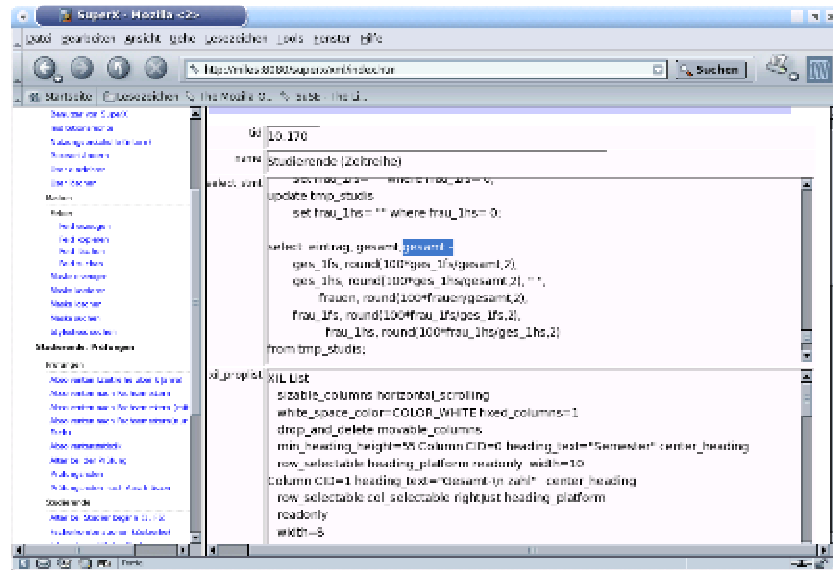
Ein kurzer Hinweis zur `xil_proplist`: Spaltenüberschriften sind von 0 aufsteigend durchnummeriert, und die Überschriften selbst können durch "\n" mit Zeilenumbrüchen versehen werden. Pro Überschrift wird ein Absatz formuliert, u.a. auch die Breite der Spalte. Alle anderen Angaben werden zur Zeit noch nicht ausgewertet<sup>iv</sup>. Wichtig ist dass eine Spalte pro Absatz definiert ist.

### 2.1.1.3.2 Änderung einer Abfrage

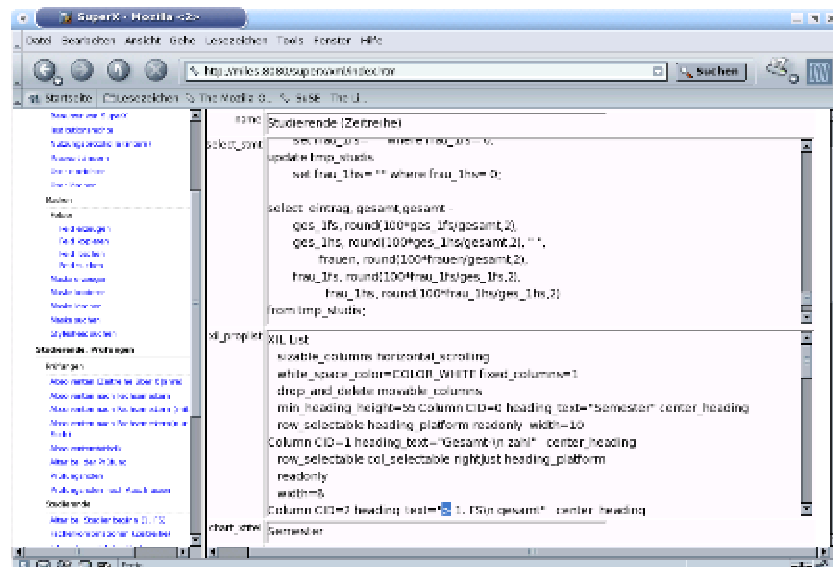
Stellen wir uns nun vor, will wollten die Abfrage dahingehend ändern, dass wir in der Ergebnistabelle statt der Studierenden im 1. FS die Studierenden im 2.-Xten Fachsemester sehen wollen. Das ist eine recht anschauliche Übung, dazu ändern wir den letzten `select`<sup>v</sup>.

Wir schreiben vor das Feld ges\_1fs den Ausdruck

"gesamt -".



Dann müssen wir die Spaltenüberschrift ändern, Spalte 2 ergänzen wir am Anfang um ein ">"-Zeichen.



Und wenn wir dann die Maske im Themenbaum neu auswählen und ablaufen lassen, erhalten wir folgendes Ergebnis:

Die dritte Spalte zeigt nicht mehr die Studierenden im 1. Fachsemester, sondern den Rest. Alle anderen Spalten haben sich nicht verändert.

The screenshot shows the SuperX 2.2 web application interface. At the top, there is a navigation bar with buttons for 'Anmelden', 'Passwort', 'Clone', 'Kopieren', 'Drucken', 'Hilfe', and 'Info'. Below this is a 'Themenauswahl' section with tabs for 'Maske' and 'Tabelle'. The main content area displays a table with the following data:

Semester	Gesamzahl	> 1. FS	1. FS	1. HS	1. HS	dar.	Frauen	1. FS	1. FS	1. HS	1. HS
	zahl	gesa...	in %	gesa...	in %	Frauen	in %	Frauen	Frauen	in %	Frauen
SS 2003	13655	13.496	1,16	63	0,46	5467	40,16	65	53,46	28	44,
WS 2002...	15049	11.943	20,64	2427	16,13	6051	40,21	1391	44,78	1106	45,
SS 2002	13661	13.126	3,92	348	2,55	5371	39,32	263	49,16	136	39,
WS 2001...	14324	11.572	19,49	2230	15,57	5681	39,66	1251	44,81	978	43,
SS 2001	12862	12.436	3,31	210	1,63	5010	38,95	203	47,65	97	46,
WS 2000...	13737	11.424	16,84	1797	13,08	5338	38,86	985	42,59	767	42,
SS 2000	13055	12.606	3,44	218	1,65	5030	38,53	235	52,24	110	50,
WS 1999...	13903	11.760	15,41	1583	11,39	5353	38,50	923	43,07	694	43,
SS 1999	13466	12.981	3,60	161	1,20	5089	37,79	252	51,96	86	53,
WS 1998...	12982	10.990	15,34	1469	11,32	4835	37,24	848	42,62	633	43,

## 2.1.2 Fazit

So viel zu unserem Einstieg in die Abfragengestaltung. Wir sehen, dass mit den Bordmitteln der Datenbank (Stored Procedures) und dem dynamischen SQL fast beliebige Statistiken generierbar sind. Die hohe Flexibilität erkaufen wir uns mit einer recht hohen Hürde beim Einstieg, außerdem ist die Arbeit nicht gerade "visuell".

Nützliche Hilfsmittel sind SQL-Generatoren wie z.B. die SQLWorkbench von Thomas Kellerer ([www.kellerer.org](http://www.kellerer.org)); bitte beachten Sie dabei, dass die Syntax der zugrunde liegenden Datenbanken berücksichtigt werden muss.

Der nächste Schritt für Sie wäre:

- Gestaltung von Abfragenlayouts im XML-Frontend
- Erkunden weiterer Feldarten- und Typen (z.B. Text- und Datumsfelder), insbes. unseren "Organigramm-Button" für Auswertungen im Bereich Haushalt, Personal etc.
- Zuordnen neuer Felder zu Masken bzw. Entfernen von Feldern
- Entwerfen von Stored Procedures

## 2.2 Erweiterte Maskenprogrammierung: Freemarker Templates

Mit SuperX 3.0 wird eine stark erweiterte Möglichkeit zur Abfragenentwicklung eingeführt.

Die OpenSource-Bibliothek *FreeMarker* ([www.freemarker.org](http://www.freemarker.org)) wird als Template-Engine eingesetzt.

Damit Sie in einer Abfrage die Freemarker Funktionalität benutzen können, muss im Kopf des `select_stmt` eine Hinweiszeile

--FREEMARKER TEMPLATE

enthalten sein.

## 2.2.1 Klassische Verarbeitung

Die einzelnen Abfragen (auch synonym Masken genannt) enthalten SQL Befehle mit Platzhaltern.

z.B. `select monat,sum(betrag) from cob_bus  
where monat=<<Monat>>`.

Auf der Maske gibt es ein Feld Monat. Vorm Abschicken des SQL wird <<Monat>> durch den gewählten Wert ersetzt.

Ausdrücke die zwischen `/*` und `*/` stehen, werden entfernt, falls kein Wert ausgewählt wurde.

Wenn man auf einer Maske z.B. optional auf einen Geldgeber einschränken kann.

Aus

z.B. `select monat,sum(betrag) from cob_bus  
where monat=<<Monat>>`.

`/* and gege=<<Geldgeber>> */`

wenn kein Geldgeber ausgewählt wurde

`select monat,sum(betrag) from cob_bus`

`where monat=1`

wenn aber ein Geldgeber ausgewählt wurde statt dessen

z.B. `select monat,sum(betrag) from cob_bus`

`where monat=1`

`and gege=3;`

Achtung:

Der Ausdruck in <<XXX>> darf nur einmal in dem optionalen Block vorkommen.

Falls er zweimal benötigt wird, muss es auf zwei Blöcke aufgeteilt werden.

z.B.

`/* and (dr in (<<Deckungsring>>)) */`

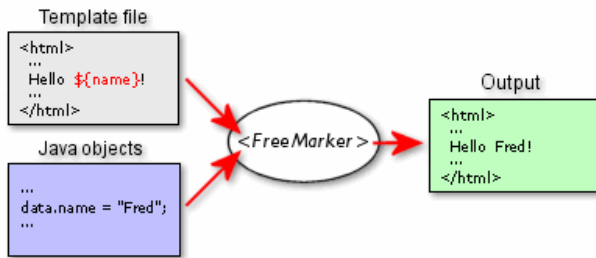
`/* or dr2 in (<<Deckungsring>>))*/`

## 2.2.2 FreeMarker Transformation

### 2.2.2.1 Übersicht

Nach der klassischen Transformation mit `generateSql` folgt ggfs. die FreeMarker Transformation.

FreeMarker transformiert eine Vorlage (template) mit Hilfe eines Datenmodells (mit Java Objekten) zu einem Ausgabertext.



Sehr oft wird es zur Erzeugung von HTML benutzt, wir produzieren statt dessen SQL.  
Die Java-Objekte im Datenmodell sind die Felder, die auf der Maske zur Auswahl stehen.

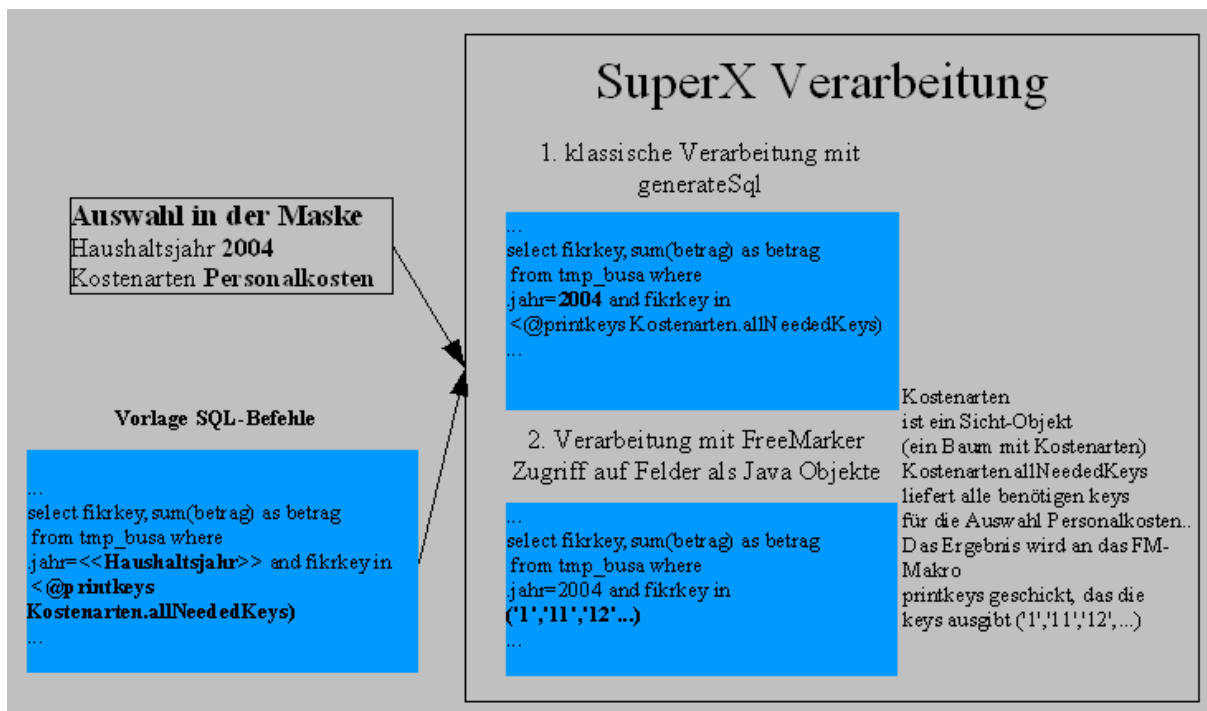
Als einfachsten Anwendungsfall könnten wir also für eine Maske mit einem Monatsfeld statt des klassischen SuperX-Tags

```
select monat,sum(betrag) from tmp_busa where monat=<<Monat>>
```

auch die FreeMarker Notation nehmen.

```
select monat,sum(betrag) from tmp_busa where monat=${Monat}
```

Ein komplexes Beispiel:



### 2.2.2.2 Programmieren mit FreeMarker

FreeMarker unterstützt praktisch alle Konzepte klassischer Programmiersprachen.

Die Tags sind HTML-ähnlich.

### 2.2.2.2.1 Zugriff auf Java-Objekte im Datenmodell

Generell geht der Zugriff auf Java-Objekte im Datenmodell mit der Notation `${varname}`.

z.B. -- ausgewählte Aggregation: `${Aggregation}`

innerhalb von FreeMarker-Befehlen muss `${}` weggelassen werden

```
<#if Aggregation="stark">
<@printkeys Kostenarten.allNeededKeys/>
```

### 2.2.2.2.2 if-Abfragen

Mit FreeMarker können Sie z.B. **if-then-Abfragen** in normales SQL einbauen, z.B. um je nach gewünschter Aggregierungsstufe einen unterschiedlichen insert zu benutzen

**if-then in normalem  
SQL**

```
<#if "<<Aggregation>>="stark">
insert into .. select ...
<#elseif "<<Aggregation>>="mittel">
insert into .. select ...
<#else>
insert into .. select ...
</#if>
```

Der klassische SuperX-Tag `<<Aggregation>>` wird vor der FreeMarker Transformation ersetzt, sodass FreeMarker effektiv zwei Strings vergleicht (if "stark"="stark").

Alternativ könnte den ausgewählten Wert des Felds Aggregation im Java-Objekt direkt ansprechen.

```
<#if Aggregation="stark">
```

Hier braucht kein `${}` um Aggregation, da wir ja schon innerhalb einer FreeMarker-Anweisung sind.

### 2.2.2.2.3 Variablen

Mit `assign` kann man eigene Variablen definieren.

z.B.

```
<#assign sortnr=0>
```

```
<#assign sortnr=sortnr+1>
```

```
insert into ... values (${sortnr},...)
```

```
<#if "<<Stichtagsbezogen>>="NEIN">
  <#assign quelltable = "sos_statistik">
<#else>
  <#assign quelltable = "sos_statistik">
</#if>
```



```
select .... from ${quelltabelle} where ...
```

<<Stichtagsbezogen>> wird von der SuperX-Transformation ersetzt, sodass Freemarker vergleicht:

```
<#if "JA"="NEIN"> oder <#if "NEIN"="NEIN">
```

Auf alle Felder einer Maske kann man neben der klassischen SuperX-Notation <<FELDNAME>> auch per Freemarker zugreifen. Es ginge z.B. auch

```
<#if Haushaltsjahr=2005>...
```

```
insert into ... select ... from ... where jahr=${Haushaltsjahr}
```

(Außerhalb von Freemarker-Anweisungen, muss die Variable mit \${ } umschlossen sein, wenn Leerzeichen oder Sonderzeichen vorkommen muss man die .vars-Notation benutzen,

z.B. "alles aufsummieren?"))

Wenn man wissen möchte, ob eine Variable mit Inhalt gefüllt ist, kann man dies mit has\_content Abfragen, z.B.

```
<#if lehr_abg?has_content >
```

Folgender Effekt ist schon mal aufgetreten:

Wenn man ein einer Abfrage z.B. schreibt

```
<#assign sortnr=sortnr+1>
```

```
insert into tmp_rs_base
```

```
(struktur,text, ch30_fach, kz_fach, fach_nr, ch35_ang_abschluss, sortnr,
```

```
...
```

dann klappt das nicht, man muss unter dem assign eine Leerzeile machen:

```
<#assign sortnr=sortnr+1>
```

```
insert into tmp_rs_base
```

```
(struktur,text, ch30_fach, kz_fach, fach_nr, ch35_ang_abschluss, sortnr,
```

### Exkurs für Fortgeschrittene: sqlvars

Manchmal wünscht man sich mit Freemarker auf Variablen zugreifen zu können, die aus der Datenbank gefüllt werden müssten, weil sie nicht als Felder auf der Maske vorkommen.

Allerdings muss die Freemarker-Transformation ja schon laufen, bevor der fertige SQL an die Datenbank geschickt wird, weil Postgres/Informix ja mit Freemarker-Befehlen nichts anfangen können.

Man kann also nicht so etwas

```
<#assign gegenname="select name from fin_geldgeber G where G.xyz=<<XYZ>>">
```

machen, weil die Variable gegenname dann einfach nur select... als String enthält, Freemarker hat mit der Datenbank keinen direkten Kontakt.

SuperX wäre aber nicht so krass, wenn es nicht Abhilfe gäbe:

man legt einen Block an

```
<sqlvars>
<!-- einfache Variable-->
<sqlvar name="fin_geldgeber_exists">
select sp_table_exists('fin_geldgeber') from xdummy
</sqlvar>

<!-- Listenvariable -- 1. Spalte key (darf nicht null sein!), 2. Spalte name-->
<sqlvar name="geldgeber">
select ggnr,ggname1 from fin_geldgeber
</sqlvar>
</sqlvars>
```

Anschließend kann man in der Abfrage mit FreeMarker auf diese aus der Datenbank gefüllten Variablen zugreifen:

```
<#if fin_geldgeber_exists=1>
insert into .. select * from fin_geldgeber;
<#else>
insert into .. select * from fin_geldgeber;
</#if>
```

Die Variable wird als einfacher Wert erkannt, weil nur eine Spalte im SQL selektiert wurde.

(Achtung: Sofern der Select mehrere Zeilen liefert, wird nur der letzte gefundene Wert hinterlegt, wenn man abfragen möchte, ob überhaupt etwas in der Datenbank gefunden wurde kann man `<#if fin_geldgeber_exists?has_content>` benutzen)

Die zweite Art von Variable (Geldgeber) wird als Liste von Items geführt und kann in der Abfrage benutzt werden z.B. mit

```
where ggnr in (
<#foreach gg in geldgeber>
${gg.key},
</#foreach>
```

Die erste per SQL eingelesene Spalte, ist das Attribut key, die zweite name.

optional kann auch noch dritte/vierte Spalte mit Strukturinfo eingelesen werden

```
<sqlvar name="geldgeber">
select ggnr,ggname1,klr_geldgeber,strukturint from fin_geldgeber
</sqlvar>
```

In den sqlvars kann auch freemarker-syntax und repository/konstanten benutzt werden, obstruses Beispiel:

```
<sqlvar name="spezial_gege">
<#if K_FIN_Quellsystem=1>
select ggnr,ggname1 from fin_geldgeber where ggnr in ${FIN_Drittmittel}
<#else>
select ggnr,ggname2 from fin_geldgeber where ggnr in ${FIN_Drittmittel}
```

```
</#if>
</sqlvar>
```

#### 2.2.2.2.4 has\_content

Wenn man wissen möchte, ob eine Variable mit Inhalt gefüllt ist, kann man dies mit has\_content Abfragen, z.B.

```
<#if lehr_abg?has_content >
```

#### 2.2.2.2.5 ForEach

FreeMarker kann nicht nur primitive Datentypen wie Strings oder Zahlen verarbeiten, sondern auch Collections. Wenn im Datenmodell eine Collection hinterlegt ist, kann man forEach benutzen.

Das sieht ungefähr so aus

```
<#foreach eineKostenart in Kostenarten.elements>
-- Auswertung für ${eineKostenart}
</#foreach>
```

Details siehe bei Sichtfeldern-Schleifen.

#### 2.2.2.2.6 For ...Next ...-Schleifen: List

FreeMarker kann auch eine For-Next-Schleife mit 1-er Schritten erzeugen.

Das sieht z.B. so aus:

**Erzeugung einer temp.  
Tabelle mit Altersin-  
tervallen  
m\_a18,...,m\_a20,...**

**ergibt nach Freemar-  
ker-Transformation**

```
create temp table tmp_aggre \
(struktur char(50),text char(200), ch30_fach char(3),sortnr
int,
<#list 0..30 as i>
m_a${18+(i*2)} decimal(7,2),
w_a${18+(i*2)} decimal(7,2),
</#list>
gesamt decimal(7,2));
create temp table tmp_aggre
(struktur char(50),text char(200), ch30_fach char(3),sortnr
int,
m_a18 decimal(7,2),
w_a18 decimal(7,2),
m_a20 decimal(7,2),
w_a20 decimal(7,2),
m_a22 decimal(7,2),
w_a22 decimal(7,2),
[...]
m_a78 decimal(7,2),
w_a78 decimal(7,2),
gesamt decimal(7,2));
```

### 2.2.2.2.7 Makros und Funktion

Es könnten auch Makros und Funktionen (die Werte zurückliefern) definiert werden.

```
<#macro macroname param1 param2>
</#macro>
```

Aufgerufen werden sie mit

```
<@makroname> bzw. <@makroname param1 param2 ..>
```

Die Reihenfolge in der Makros definiert werden spielt keine Rolle.

Einige Makros für Datenbankunabhängigkeit (SQL lingua franca) und allgemeine Makros sind in der Tabelle fm\_templates hinterlegt.

Diese können mit

```
<#include "xx"/> eingebunden werden.
```

### 2.2.2.3 Special tricks

Kostenarten name = xxx-Name

nur Namen ausgeben:

```
'${eineKostenart.name[(eineKostenart.name?index_of('-')+1)..(eineKostenart.name?length-1)]}'
```

```
<#assign maxEbene=99>
<#if "<<Filter bis Ebene>>"!="">
/* <#assign maxEbene=<<Filter bis Ebene>> */
</#if>
```

```
<<SQL>> /* execute procedure sp_user_orga_child(<<UserID>>,mdy (1, 1,year(<<Organigramm-
Stand>>)),<<Organigramm-Sicht>>,<<Institution>>,<<erlaubt>>); --notinxmlfrontend */
/* execute procedure sp_user_orga(<<UserID>>,mdy (1, 1,year(<<Organigramm-Stand>>)),<<Organigramm-
Sicht>>); create temp table tmp_ch110_institut (ch110_institut char(10)); insert into
tmp_ch110_institut select key_apnr from tmp_organigramm; drop table tmp_organigramm; --notinapplet */
select distinct fb, fb || " - " || ktobez from mbs_projekte_konto where l=1 and inst_nr in (select
ch110_institut from tmp_ch110_institut) /* and dr in (<<Deckungsring>>) */ /* and jahr = <<Haushalts-
jahr>> */ /* and kap in (<<Kapitel>>) */ order by 2;
```

Achtung - Absatzmarke scheint auch wichtig zu sein

### 2.2.2.4 SQL-Lingua Franca

Um Abfragen nicht separat für Informix/Postgres entwickeln zu müssen, gibt es FreeMarker SQL-Lingua Franca<sup>vi</sup> Makros.

Einfaches Beispiel:

Aufruf einer Prozedur

Informix execute procedure sp_proc();
Postgres select sp_proc();

FreeMarker <b>&lt;@procedure sp_proc/&gt;</b>	wird transformiert →	Informix execute procedure sp_proc();
		Postgres select sp_proc();

procedure ist ein FreeMarker Makro, das definiert ist in der Tabelle fm\_templates, id "SQL\_lingua franca"

**<@informixnolog/>**

wenn Informix als Datenbanksystem benutzt wird, wird der Zusatz *with no log* ausgegeben.

**selectintotmp**

Informix und Postgres unterscheiden sich darin, wie ein select into temp table aufgebaut ist.

Beispiel

Informix

*select key,sum(betrag) from cob\_busa where ... group by .... **into temp tmp\_busa;***

und

Postgres

*select key,sum(betrag) **into temp tmp\_busa** from cob\_busa where ... group by ...;*

Um dieses datenbankunabhängig zu halten, gibt es das Makro selectintotmp mit den parametern select (die Spalten), source (Quelltabelle/n) und target (Zieltabelle).

Eine gegebenenfalls nötige where-Bedingung und group by muss als "Body" innerhalb des selectintotmp-Aufrufs angegeben werden.

Die obigen Beispiele können mit FreeMarker mit folgendem Makroaufruf automatisch erzeugt werden.

```
<@selectintotmp select="key,sum(betrag)" source="cob_busa" target="tmp_busa">
where ... group by ..
```

```
</@selectintotmp>
```

### 2.2.2.5 Allgemeine FM-Makros/Funktionen

Es gibt eine Reihe von allgemeinen Makros, die für alle Abfragen interessant sind.

Standardmäßig werden Kommentare von der klassischen SuperX-Verarbeitung (generateSql) gelöscht, damit nicht soviel an die DB geschickt wird.

Wenn man aber bei Entwicklungszwecken noch Kommentare drin haben will, kann man das Makro addcomment benutzen.

Da FreeMarker erst nach dem klassischen generateSql ausgeführt wird, kann man Kommentare wieder einfügen.

```
<@addcomment comment="Hier wirds interessant"/>
```

#### **printkeys**

Druckt die Schlüssel aus.

```
<@printkeys Kostenarten.allNeededKeys/>
```

schreibt z.B. wenn Personalkosten ausgewählt wurde ('1','11','12',...)

Entwicklungsmöglichkeit:

Man könnte bei Bedarf gewisse "künstliche" Schlüssel rausfiltern.

### 2.2.2.6 Spezielle Möglichkeiten bei Sicht-Feldern

Bei Sicht-Feldern (Feldart 12) gibt es besondere Möglichkeiten. Mittels Java-Reflection kann FreeMarker auf Methoden der Objekte im Datenmodell zugreifen.

Bei Sichtfeldern wie Org. Einheit oder Kostenart sind folgende Methoden interessant.

#### 2.2.2.6.1 allNeededKeys – für temporäre Datentabellen

Diese Methode liefert alle benötigten Schlüssel.

Wenn bei Org. Einheit oder Kostenart nichts ausgewählt wurde, werden einfach alle im Baum vorhandenen Schlüssel geliefert.

Wenn z.B. Personalkosten ausgewählt wurde, wird nur der Schlüssel von Personalkosten ('1') und dessen Unterknoten (z.B. '11','12') geliefert.

Dafür wird noch das allgemeine Makro printkeys benutzt.

```
<@printkeys Kostenarten.allNeededKeys/>
```

Beispiel für Erstellung einer temporären Datentabelle

```
execute procedure sp_user_orga_child
  (<<UserID>>,<<Organigramm-Stand>>,0,<<Institution>>, <<erlaubt>>);
Create temp table tmp_erg (fikir varchar(200), betrag decimal (14,2)) with no log;
```

```
select fikirkey,sum(betrag) as betrag from cob_busa B,tmp_ch110_institut T where
  B.ch110_institut=T.ch110_institut and
  B.jahr=<<Haushaltsjahr>> and fikirkey in
  <@printkeys Kostenarten.allNeededKeys />
group by fikirkey
into temp tmp_busa;
```

(Statt Benutzung der Prozedur sp\_user\_orga\_child könnte man analog verwenden:  
 where B.110\_institut in <@printkeys Institution.allNeededKeys/>  
 Ggfs. versteckte Knoten werden hier mit ausgegeben.  
 Bei Kostenstellen-Feldern werden nur erlaubte Einträge ausgegeben.

#### 2.2.2.6.2 keysToRoot – für Verteilschritte

Verteilschritte sind ein ungewöhnliches Konzept.

- 1
- 2
- 3

Schritt 3 ist die Summe aus 1-3.

In Abfragen wird <@printkeys Verteilschritt.keysToRoot/> benutzt  
 (Ab Kernmodul 3.0rc3)

#### 2.2.2.6.3 elements– für Schleife über ausgewählte Knoten

Diese Methode liefert eine Collection, entweder über alle Knoten im Sichtbaum oder nur über einen ausgewählten Knoten und deren Kinder.

Beispiel:

```
<#foreach eineKostenart in Kostenarten.elements>
-- Auswertung für ${eineKostenart}

</#foreach>
```

elements liefert die Knoten genau in der Reihenfolge, in der sie auch im Baum sind.

Alternativ kann man breadthFirstElements oder depthFirstElements angeben.

Dann wird beim Baum zunächst in die Breite/Tiefe gegangen.

**Wichtig:**

Für eine foreach-Schleife werden auch bei Kostenstellen-Feldern bei eingeschränkten Usern immer alle Knoten ausgegeben

z.B. nur Rechte auf 11 und 13

root-Hochschule (Auswahl)

fak-Fakultäten (Auswahl)

1-fak1 (Auswahl)

11-Institut 1

13 – Institut 3

Es werden alle Knoten durchlaufen, weil (Teil)summenzeilen interessant sein können.

Anders ist es bei Berechnung (Methode subkeys) da werden nur die tatsächlich erlaubten Schlüssel ausgegeben.

**2.2.2.6.4 Zugriff auf einzelne Knoten im Baum**

Im Rahmen einer forEach Schleife bekommt man Zugriff auf einzelne Elemente eines Sichtenbaums.

Für die einzelnen Knoten kann FreeMarker wiederum mittels Java-Reflection auf bestimmte Methoden zugreifen.

**id, name** – Zugriff auf den Schlüssel und den Namen des Knotens

Insert into tmp\_erg (fikt , betrag )

SELECT "\${eineKostenart.id}" || " " || "\${eineKostenart.name}", sum(betrag)

FROM tmp\_busa

...

**subkeys** – liefert eine Liste mit dem Schlüssel des aktuellen Knotens (z.B. Personalkosten '1') und aller seiner Unterknoten (z.B. '11','12') (auch von versteckten Knoten!)

Insert into tmp\_erg (fikt , betrag )

SELECT "\${eineKostenart.id}" || " " || "\${eineKostenart.name}", sum(betrag)

FROM tmp\_busa

where fiktkey in <@printkeys **eineKostenart.subkeys**>

group by 1 ;

Bei Kostenstellen-Sichten werden nur die erlaubten Knoten ausgegeben,

z.B.

root-Hochschule (Auswahl)

fak-Fakultäten (Auswahl)



1-fak1 (Auswahl)

11-Institut 1

13 – Institut 3

wenn nichts ausgewählt wurde, root, fak, oder fak1 wird trotzdem nur 11,13 als subkeys ausgegeben, weil nur die selbst erlaubt sind. -> bei foreach (Methode elements ist es anders root, fak, fak1 werden auch mit durchlaufen, weil (Teil)summenzeilen interessant sein können

**strukturInt,strukturStr** - beschreibt Art oder Struktur des Knotens

Beispiel: orgstruktur im Organigramm beschreibt, ob ein Knoten eine Lehreinheit oder ein Fachbereich ist (20 bzw. 30)

Diese Strukturinformation ist im Knoten hinterlegt, sofern beim Einlesen der Sicht an Position 4 und/oder 5 etwas angegeben wurde

(z.B. select name,key\_apnr,parent,orgstruktur from organigramm).

Sie kann z.B. für if-Abfragen zur Aggregation benutzt werden.

```
<#foreach eineInstitution in Institutionen>
  <#if Aggregation="stark" and eineInstitution.strukturInt=30>
    ...
  <#else>
    ..
  </#if>
</#foreach>
```

**nodeattrib** – ein Knotenattribut

wenn null oder 0: Knoten ganz normal, 1 versteckt, 2 nicht selektierbar im Baum

### 2.2.2.7 Grundgerüst für neue Abfragen

Voraussetzung:

Feld über das Schleife laufen soll, muss eine Sicht sein (Feldart 12)

Schritt	Beispiel
1. temporäre Datentabellen erstellen	<pre>&lt;@selectintotmp select="ch30_fach,..." source="sos_statistik S" target="tmp_sosstatistik"&gt;   where &lt;&lt;Köpfe oder Fälle&gt;&gt; and &lt;&lt;Hörerstatus&gt;&gt; and ... and ch30_fach in &lt;@printkeys Fächer.allNeededKeys/&gt;</pre>

	<pre> &lt;/@selectintotmp&gt; &lt;@informixnolog/&gt;; </pre>
2. Schleife über alle gewünschten Knoten	<pre> --Schleife, über jede Kostenart im ausgewählten Kostenarten- Baum, Reihenfolge genau wie im Baum &lt;#foreach eineKostenart in Kostenarten.elements&gt; --mit if-Anweisungen können ggfs. einzelne Einträge übersprungen werden (z.B. wegen Aggregierungsauswahl) Insert into tmp_erg (fikt , betrag ) SELECT  '\${eineKostenart.id}':char(10)    ' '    '\${eineKostenart.name}':char(100), sum(betrag) FROM tmp_busa where fiktkey in \${eineKostenart.subkeys} --liefert nötige Schlüssel als ('1','12','13'..) für aktuelle Kostenart group by 1 ; &lt;/#foreach&gt; </pre>
3. ggfs Weiterverarbeitung	<pre> update tmp_erg set ..... </pre>
4. Abschluss-Select	<pre> select * from tmp_erg; </pre>

### 2.2.3 Datenbankunabhängigkeit

Grundlage für die Datenbankunabhängigkeit (Informix/Postgres) von SuperX-Abfragen sind die Free-Marker Makros SQL Lingua Franca (s.o.).

Zusätzlich sollte man auf Folgendes achten.

#### entwickeln mit Postgres

Da Postgres bei einigen Dingen strenger ist, sollte man mit Postgres entwickeln, wo möglich.

#### einfache Hochkommata haben Vorrang

```

SELECT  '${eineKostenart.id}':char(10), sum(betrag)
FROM tmp_busa ...

```

Informix würde auch "\${eineKostenart.id}" verstehen, Postgres würde dies jedoch für einen festen Spaltennamen halten und meckern, dass es keine solche Spalte gibt.

#### Casting benutzen

Postgres muss öfter mittels Casting über den Datentyp informiert werden.

```

insert into ...

```

```

select 'Summe':char(100), .... from ... group by 1,2,3;

```

Informix braucht das nicht unbedingt, Postgres meldet bei fehlendem Casting gerne:

ERROR: Unable to identify an ordering operator '<' for type "unknown"

Use an explicit ordering operator or modify the query

Andersherum ist es wichtig, bei unions auch null zu casten.

```
select x,y,z from test
```

```
union
```

```
select x, null::char(10),z from test2
```

Postgres kommt ohne Casting klar, weil es aus dem ersten Select den Datentyp erkennt. Informix meldet demgegenüber nur Syntax Error.

## 2.2.4 Zugriff auf Konstantentabelle und Hochschulinfo

Die Konstantentabelle steht auch komplett als FreeMarker-Variablen zur Verfügung. Damit es keine etwaigen Überschneidungsprobleme mit Feldernamen gibt, wird ein K\_ vorangestellt.

Die COB\_Version könnte also z.B. so abgefragt werden

```
<#if K_COB_Version>7 >
```

```
...
```

```
<#else>
```

```
...
```

```
</#if>
```

Die ersten Einträge der Tabelle hochschulinfo sind ebenfalls verfügbar als

K\_Name, K\_Adresse, K\_hs\_nr und K\_Kapitel.

## 2.2.5 Sx\_repository

In der Tabelle sx\_repository können Einträge hinterlegt werden, die in allen Abfragen zur Verfügung stehen, z.B. kann dort ein SQL hinterlegt werden, der bestimmt, wie sich ein Professor in SVA definiert (z.B. bvl=30000 and dienstbez like „Prof“). Die Tabelle ist folgendermaßen aufgebaut:

tid	serial		1
id	bpchar(200)	Sollte mit der Modulbezeichnung anfangen und keine Leerzeichen enthalten	SVA_Professor
content	text(-1)	der Inhalt, der in den Abfragen eingesetzt werden soll	bvl=30000 and dienstbez='prof'
caption	bpchar(200)	Bezeichnung, die ggfs. auch auf dem Bildschirm mit ausgegeben werden soll	Professoren
comment	text(-1)		
version	int2		

art	bpchar(200)		
sachgebiete_id	int4		
sort1	int4		
sort2	int4		
sort3	int4		
gueltig_seit			
gueltig_bis			

Die Repository wird im Applet bei der Anmeldung mit übermittelt, im XML-Frontend wird es gecacht. (Wenn Änderungen gemacht werden, muss einmal im SuperXManager der Server-Cache aktualisiert werden).

In den Abfragen kann man die Repository Einträge z.B. folgendermaßen verwenden

insert into tmp\_erg (bezeichnung, wert)

```
select "Professoren", sum(value) from xxx where ${SVA_Professor}
```

anstatt den Beschreibungstext fest anzugeben, kann man auch die caption des Eintrags benutzen.

```
select "${SVA_Professor.caption}", sum(value) from xxx where ${SVA_Professor}
```

Wenn es nur einen Eintrag für eine ID gibt, reicht die beschriebene Vorgehensweise. Falls es aber für eine ID mehrere Einträge mit unterschiedlichen Gültigkeitszeiträumen gibt, schreibt man z.B.

```
select "Sachmittel", sum(value) from xxx where ${FIN_Sachmittel("1.1.<<Rechnungsjahr>>")}
```

dann wird der erste gefundene FIN\_Sachmittel Eintrag ausgegeben, der zum angegebenen Datum gültig ist. Dabei kann bei Bedarf auch wieder auf einzelne Attribute zugegriffen werden, also z.B.

```
select "${FIN_Sachmittel("1.1.<<Rechnungsjahr>>").caption}", sum(value) from xxx where  
${FIN_Sachmittel("1.1.<<Rechnungsjahr>>")}
```

## 2.2.6 Abfragen mit variabler Spaltenzahl

Es ist nun mit FreeMarker auch möglich, Abfragen mit einer variablen Anzahl von Ergebnisspalten zu erzeugen. Dazu muss mit Freemarker an zwei Stellen Variabilität eingebracht werden:

- im SQL der Maske und
- in der XIL-Proplist, die die Definition der Ergebnisspalten enthält.

Einfaches Beispiel:

Im FIN-Modul gibt es zwei Quellsysteme (MBS und KAHIKA). Je nach Quellsystem soll in einer Abfrage unterschiedliche Spalten erscheinen. Das Quellsystem kann über die Konstante FIN\_QUELLSYSTEM ermittelt werden (Zugriff über FreeMarker als K\_FIN\_QUELLSYSTEM, 1=MBS, 2=KAHIKA). Am Ende des Masken-SQLs kann beispielweise stehen:

```

<#if K_FIN_Quellsystem=2>
select eintrag, hhans,reste, bewegungen, fest, verfuegbar, verfuegbar-fest from
tmp_erg order by dr,sortnr,tit;
<#else>
select eintrag, hhans,reste,akts_ein, sperr, angeordne-
ta,offsoll_a,angeordnete,offsoll_e, fest, verfuegbar, verfuegbar-fest from tmp_erg
order by dr,sortnr,tit;
</#if>

```

Die definierten Spaltenüberschriften in der XIL-Proplist müssen ebenfalls variabel angelegt werden:

```

--freemarker template
XIL List
  drop_and_delete movable_columns sizable_columns horizontal_scrolling
  white_space_color=COLOR_WHITE fixed_columns=1
  min_heading_height=35
Column CID=0 heading_text="Titel / DR" center_heading
  row_selectable col_selectable heading_platform readonly
  width=20 text_size=8 explanation="@@@fin_titel_dr@@@"
Column CID=0 heading_text="Zuweisungen" center_heading
  row_selectable col_selectable heading_platform readonly
  width=15 text_size=8 explanation="@@@fin_zuweisungen@@@"
Column CID=0 heading_text="Reste" center_heading
  row_selectable col_selectable heading_platform readonly
  width=15 text_size=8 explanation="@@@fin_reste@@@"
<#if K_FIN_Quellsystem=2>
  Column CID=0 heading_text="Bewegungen" center_heading
  row_selectable col_selectable heading_platform readonly
  width=15 text_size=80 explanation="@@@fin_bewegungen@@@"
<#else>
Column CID=0 heading_text="Akt.Soll" center_heading
  row_selectable col_selectable heading_platform readonly
  width=15 text_size=8 explanation="@@@fin_akt_soll@@@"
  ...
</#if>
  Column CID=1 heading_text="Festgelegt" center_heading
  row_selectable col_selectable heading_platform readonly
  width=15 text_size=80 explanation="@@@fin_festgelegt@@@"

```

Grundprinzip ist also, dass mit FreeMarker die Variabilität in den SQL und die XIL-Proplist gebracht werden muss.

Ein komplexes Beispiel:

In einer Abfrage soll ein variable Anzahl von Lehreinheiten in den Spalten erscheinen, diese können nicht direkt aus einem Sicht-Feld Lehreinheiten ermittelt werden (in FreeMarker wäre dann z.B. Zugriff über `<#foreach lehrein in Lehreinheiten>` möglich). Stattdessen wird die Gesamtliste aller gewünschten Lehreinheiten über die Felder *Semester* und *Anbietende Lehreinheit* definiert. Damit Freemarker vor der eigentlichen Transformation noch eine passende aus der Datenbank gefüllt Variable erhält, wird am Anfang der Abfrage eine sqlvar namens *lehr\_abg* angelegt (vergl. Abschnitt zu [Variabeln](#) (S. 24)).

```

--Freemarker Template
<#include "SQL_lingua_franca"/>
<#include "SuperX_general"/>
<sqlvars>

```

```

<sqlvar name="lehr_abg">
SELECT distinct L.key_apnr,
           L.drucktext, L.name as strukturstr
from gang_k_lehr_hs L, gang_k_semester S
  where
1=1
/* and L.key_apnr in(<<Anbietende Lehreinhh.>>) */
/* and S.tid = <<Semester>> */
  and S.sem_beginn between L.gueltig_seit and L.gueltig_bis
order by 2;
</sqlvar>
</sqlvars>

```

Anschließend wird die Ergebnistabelle variabel erzeugt (has\_content prüft, ob überhaupt Lehreinheiten gefunden wurden).

```

create temp table tmp_gang_cnw2 (
ord integer,
tid integer,
semester_von integer,
sem_beginn date,
lehr_abg character(10),
lehr_empf character(10),
lehr_empf_str char(255),
lehr_empf_sort char(255),
stg_empf_str nchar(255)
<#assign i=0 />
<#if lehr_abg?has_content >
<#foreach row_gang in lehr_abg>
<#assign i=i+1 />
, lehr_${i} decimal(5,2)
</#foreach>
</#if>
)

```

An späterer Stelle können variabel updates auf die Ergebnistabelle gemacht werden, damit kann mit .key auf den Schlüssel der Lehreinheit zugegriffen werden.

```

<#assign i=0 />
<#if lehr_abg?has_content >
<#foreach row_gang in lehr_abg>
<#assign i=i+1 />
update tmp_gang_cnw2 set lehr_${i} = (select sum( C.ca_wert)
from tmp_gang_cnw C
where C.lehr_abg=tmp_gang_cnw2.lehr_abg
and C.lehr_empf=tmp_gang_cnw2.lehr_empf
and C.tid=tmp_gang_cnw2.tid
and tmp_gang_cnw2.lehr_abg='${row_gang.key}')
where ord=2;
</#foreach>

```

```
</#if>
```

Zum Schluss könnte man einfach `select * from tmp_xx;` machen, oder sicherheitshalber besser:

```
select ord as ebene, stg_empf_str
<#assign i=0 />
<#if lehr_abg?has_content >
<#foreach row_gang in lehr_abg>
<#assign i=i+1 />
, lehr_${i}
</#foreach>
</#if>
from tmp_gang_cnw4
```

Schließlich muss nur noch die XIL-Proplist variabel erzeugt werden, dabei kann mit `.name` auf den Namen der Lehreinheit zugegriffen werden.

```
XIL List
  sizable_columns horizontal_scrolling
  white_space_color=COLOR_WHITE fixed_columns=1
  min_heading_height=35
Column CID=0 heading_text="Ebene" explanation="" center_heading
  row_selectable col_selectable rightJust heading_platform readonly
  width=30
Column CID=1 heading_text="NachfragendenLehreinheit\\nStudiengang" explanation="" center_heading
  row_selectable col_selectable rightJust heading_platform readonly
  width=30
<#assign i=1 />
<#if lehr_abg?has_content >
<#foreach row_gang in lehr_abg>
<#assign i=i+1 />
Column CID=${i} heading_text="${row_gang.name}" explanation="" center_heading
  row_selectable col_selectable rightJust heading_platform readonly
  width=20
</#foreach>
</#if>
```

Die volle Flexibilität ist im XML-Frontend gegeben. Da sich das Applet Spaltendefinitionen merkt, kann es Unstimmigkeiten geben, wenn die Spaltenzahl von Auswahlfeldern der Maske abhängt.

## 2.2.7 Tooleinsatz

### 2.2.7.1 Jedit

Wenn intensiv mit FreeMarker gearbeitet wird, ist es angenehm mit Jedit zu arbeiten.

Dazu speichert man das `select_stmt` der betreffenden Abfrage in einer Datei, z.B. `11300.sql`.

Wenn man diese mit Jedit öffnet, hat man schon mal SQL-Syntax Highlighting, man kann aber bei Bedarf auch Freemarker-Syntax Highlighting einschalten (Menü Utilities / Buffer Options / Edit Mode -> Freemarker).

Sehr angenehm ist auch die Folding Funktion (s.u.).

Um ein geändertes Skript komfortabel einzuspielen, kann man das Skript `sx_masken_sql_update.x` benutzen.

### 2.2.7.1.1 FreeMarker-Syntax Highlighting

Menü Utilities / Buffer Options / Edit Mode -> Freemarker

### 2.2.7.1.2 Folding

Mit dem Jedit-PluginManager das Plugin "ConfigurableFoldHandler" installieren (klappte mit Jedit 4.2, bei 4.3 teilweise Probleme)

Einzelne Abschnitt mit `--start` und `--end` Kommentaren versehen

z.B.

`--start Schleife`

`<#foreach ..`

`...`

`</#foreach>`

`--end Schleife`

Menü Plugins/ConfigurableFoldHandler.

Use default ausschalten,

statt dessen

`--start` und `--end` als Fold String eingeben.

Menü Utilities/Buffer Options/Folding Mode: custom

und dann

Menü Folding / Collapse All Folds

### 2.2.7.2 `sx_masken_sql_update.x`

Wenn man das `select_stmt` einer Abfrage in einer eigenen Datei bearbeitet (z.B. mit Jedit), kann man es mit dem Skript `sx_masken_sql_update.x` komfortabel in die Datenbank einspielen.

Die Syntax ist einfach

`sx_masken_sql_update.x Maskennummer Pfad/zur/Datei.sql`

`sx_masken_sql_update.x 11300 /home/superx/entwicklung/11300.sql`



## 2.3 Abfragenentwurf mit SuperX-Sichten

In SuperX können bei Auswahldialogen verschiedene Sichten angeboten werden.

Alternative Hierarchien bzw. Auswertungshierarchien aus COB werden automatisiert übernommen. Jede Hierarchie bekommt einen Eintrag in der Sichtentabelle.

Die Sichten werden in der folgenden Tabelle definiert:

Spalte	Datentyp	Beschreibung	Beispiel	Beispiel 2
tid	integer (serial)	eindeutiger Identifier, der beim Einlesen und auch zur Rechtevergabe benutzt wird Die tid 0 bis 9 sind für interne Sichten reserviert.	0	100
parent	integer	Uum zukünftig evtl. Hierarchien von Sichten abzubilden		
system info_id	integer	querverweis zur systeminfo 0 für allgemein	0	10
art	char	Art der Sicht. Bezeichnung sollte auf –Sicht enden, dann wird auch xxx-Sicht und xxx-Stand im Masken-SQL ersetzt	Organigramm-Sicht	Kosten-/Erlösarten-Sicht
type	integer	10 für reguläre Sichten 20 für alternative Hierarchien		
name	char	Name, der auf dem Bildschirm bei der Auswahl erscheint	Organisatorische Sicht	Standardsicht-Kostenarten
name_intern	char	interner eindeutiger Name, wenn der interne Name mit memtext beginnt, sollte die Sicht nicht eigenständig geändert werden		
beschreibung	char	ausführliche Beschreibung der Sicht		
stand button	integer	soll man bei einer Sicht, den Stand ändern können muss für eine Sichtart (Organigramm-Sicht) einheitlich sein	1	1
label	integer	kann zur Charakterisierung einer Sicht benutzt werden in Duisburg-Essen intern für altes Feld lehre benutzt 0 = alle Org.Einheiten 1 = Nur Lehre anzeigen; Institutionen im Bereich Lehre gefiltert nach user_institution 2 = Nur Lehre anzeigen; Institutionen im Bereich Lehre NICHT gefiltert wird von Prozeduren (sp_user_org,sp_user_orga_child) noch unterstützt, intern aber auf neue Variante Label und User-Rechte umgesetzt	0	0
user_rechte	integer	sollen beim Aufbau der Hierarchie (vergl. Spalte quelle) Userrechte berücksichtigt werden, wird bisher nur von Duisburg-Essen benutzt	1	1
rechtequelle	char(255)	für zukünftig erweiterte Rechteverwaltung		

sesamkey	char(100)	für zukünftig erweiterte Rechteverwaltung		
quelle	text	SQL zum Aufbau der Hierarchie vergl. ausführliche Beschreibung unten	sp_user_orga(<<UserID>>,<<Stand>>,<<Sicht>>); select name,key,parent,lehre,erlaubt from tmp_organigramm;drop table tmp_organigramm;	sp_cob_fikr_hier(<<UserID>>,<<Stand>>,<<Sicht>>);select name,key,parent from tmp_hier;drop table tmp_hier;
attribut1	char	bei Bedarf noch Attribute der Sicht hinterlegt werden, auf die man bei Bedarf Einschränkungen fahren kann		
attribut2	char			
attribut3	integer			
attribut4	integer			
alt_quelle	char	Tabelle mit Infos zu altn. Hierarchie		
alt_hier_id		id der alt.Hier in angegebenen Quelltable null bei reguläre Hierarchie		
treecfg-table	char	Tabelle mit Infos zu TreeView aus Cob		
treecfgid		id des benutzen trees aus Cob-Tabelle trees null bei regulärer Hierarchie		
implied- Tags	text	future use spezifische Tags die in select_stmts ersetzt werden. Syntax <<Tagname>>=value..l.. Diese Spalte wird von Memtext gepflegt und darf nur von Memtext angepasst werden.		
userTags	text	future use: Falls Sie selbst spezielle Tags für eine Sicht hinterlegen wollen, können Sie das analog zur impliedTags-Spalte hier tun.		
xmlmaxentries	integer	wenn ein Wert angegeben ist, werden maximal die Anzahl von Einträgen in Comboboxen im Xml-Frontend angezeigt. Es werden soviele Einträge auf unteren Ebenen ausgeblendet wie nötig.		
sortnr	integer	kann für Sortierungen benutzt werden		
aktiv	integer	über das Feld aktiv können bei Bedarf Aktivierung/Deaktivierung vorgenommen werden, wenn man z.B. im Felderinfo relation schreibt <<SQL>>select tid from sichten where art='Organigramm-Sicht' and aktiv=1		
guel- tig_seit	date	Zukünftig für Gültigkeitszeiträume von Sichten		
guel- tig_bis	date			

### Quelle

Sql der ausgeführt werden soll. <<SQL>> am Anfang ist optional.

Wenn der SQL mit `sp_` anfängt, wird davor je nach Datenbankserver `execute procedure` oder `select` gesetzt. Für alle Sichten wird erwartet, dass mindestens drei Felder *name*, *key* und *parent* geliefert werden. (der konkrete Name der Spalten ist irrelevant).

Anschließend können optional noch Strukturinformationen<sup>vii</sup> folgen (entweder Integer oder String).

Beispiel:

```
sp_fin_inst_hier(<<UserID>>,<<Stand>>,<<Sicht>>);select name,key,parent,strukturint from tmp_hier
order by name; drop table tmp_hier;
```

strukturint (gefüllt aus `fin_inst.orgstruktur`) gibt mit Zahlen an, ob ein Eintrag eine Lehreinheit oder ein Fachbereich ist.

Auf diese Information kann später in Abfragen mit FreeMarker zugegriffen werden.

Beispiel

```
select
<#foreach eineInstitution in Institutionen>
  <#if Aggregation="stark" and eineInstitution.strukturInt=30>
    ...
  <#else>
    ..
  </#if>
</#foreach>
```

Knoten verstecken oder nicht-selektierbar machen.

### 2.3.1 Einträge verstecken oder nicht-selektierbar machen

Es kann gewünscht sein, dass Einträge versteckt hinterlegt werden, wenn z.B. immer bei Auswahl eines Instituts 0405 auch intern dessen alte Nummer 0205 berücksichtigt werden soll.

Der User soll in SuperX nur 0405 sehen und auswählen können. Bei Berechnung durch die Datenbank soll aber 0405 und 0205 berücksichtigt werden.

Um einen solchen Effekt zu erreichen, muss einzelnen Einträgen für die Sicht der Wert `nodeattrib=1` gegeben werden. Das geschieht, indem man den Quell-SQL der Sicht (s.o.) erweitert, um ein *explizit benanntes* Feld `nodeattrib`, z.B.

```
sp_fin_inst_hier(<<UserID>>,<<Stand>>,<<Sicht>>);select name,key,parent,strukturint,nodeattrib from
tmp_hier order by name; drop table tmp_hier;
```

Die Position von `nodeattrib` muss nach `name`, `key` und `parent` liegen, ist ansonsten aber irrelevant. Entscheidend ist die Spaltenbezeichnung `nodeattrib`. Sichertgestellt werden muss auch, dass die Hierarchie stimmig ist, also dass der Eintrag mit dem Schlüssel 0205 als Parent den Wert 0405 hat.

Noch ein Beispiel für eine spezielle Kostenartensicht, bei der Kostenarten mit langen Schlüsseln versteckt werden sollen.

```
select lbez,key,ueberg,0 as nodeattrib from cob_fikr where len(key)<=6 union
select lbez,key,ueberg,1 as nodeattrib from cob_fikr where len(key)>6
```

Weiterhin kann es gewünscht sein, dass man in einer Sicht nur spezielle Einträge auswählen kann. So können man eine Lehreinheitsauswahl vielleicht so darstellen:

Uni XY

- Fak 1
  - Lehreinheit A
  - Lehreinheit B
- Fak 2
  - Lehreinheit C
  - Lehreinheit D

Die Kategorisierung Fak 1/2 dient der Klarheit der Darstellung, für eine Abfrage soll aber vielleicht **nur** eine konkrete Lehreinheit auswählbar sein. In dem Fall kann wie oben das nodeattrib 2 vergeben werden. Dadurch wird ein Eintrag als nicht-selektierbar markiert.

### 2.3.2 User-/Gruppenrechte

Außer Administratoren dürfen normale User zunächst einmal keine der nicht-internen Sichten benutzen.

In den Tabellen user\_sichten und user\_sichtarten kann hinterlegt werden, dass ein User eine einzelne Sicht (z.B. name\_intern Kostenstellen-Hauptsicht) oder eine ganze Sichtart (z.B. Kostenarten) benutzen darf.

Für Gruppen gibt es analog die Tabellen group\_sichten und group\_sichtarten.

Außerdem sachgeb\_sichten und sachgeb\_sichtarten, wenn Leute die ganzes Sachgebiete sehen dürfen auch Sicht sehen sollen

Die Pflege dieser Tabellen kann bequem über Administrationsformular im XML-Frontend vorgenommen werden.

Achtung:

Nach Änderungen muss der Server-Cache aktualisiert werden (<http://superx-rechner/superx/servlet/SuperXManager>) und der User muss sich neu anmelden.

Alternativ könnte auch Tomcat neu gestartet werden.

### 2.3.3 Benutzung der Sichten in Masken

In Felderinfo muss für das Feld (Org. Einheit,Fächer,Kostenstelle,Kostenart,...) als Art 12 eingetragen werden.

Im Feld Relation muss ein Select stehen, der die Tids der gewünschten Sichten zurückliefert

z.B.

```
<<SQL>>select tid,type,name from sichten where art='Organigramm-Sicht' and aktiv=1 order by ty-
pe,name
```

(Die aktiv-Einschränkung ist praktisch für Entwicklungs- und Wartungszwecke, weil man dann leicht einzelne Sichten de/aktivieren kann - Sortierung nach type, damit erst Hauptsichten und dann alternative Hierarchien angezeigt werden, dann nach Name)

Der select sollte **alle Sichtentids** liefern, die ein Administrator sehen darf.

Wenn ein User für einzelne Sichten keine Berechtigung hat, filtert der Server diese automatisch raus.

Bei einem Sichtenfeld werden Tags bezogen auf Feldname und Sichtenart ersetzt, z.B. beim Feld Org. Einheit sowohl <<Org. Einheit-Sicht>> und <<Org. Einheit-Stand>> als auch <<Organigramm-Sicht>> und <<Organigramm-Stand>>.

Empfehlenswert ist aber insbesondere bei Sichten, die Benutzung der FreeMarker-Technologie.

Wenn diese nicht benutzt wird, muss bei Abfragen mit Organigramm-Sichten ggfs. das select\_stmt in Maskeninfo angepasst werden.

Beispiel:

```
execute procedure sp_user_organ_child
  (<<UserID>>,<<Organigramm-Stand>>,<<Organigramm-Sicht>>,<<Org. Einheit>> , <<erlaubt>>)
```

### 2.3.4 Alt. Hierarchien aus CoB

Es werden Hauptsichten (Type 10) für Kostenstellen, Kosten-/Erlösarten und Kostenträger angelegt.

Außerdem für jede Auswertungshierarchie einen Eintrag mit Type 20. Im Feld quelle muss die Tabelle angegeben sein, in der die alt. Hierarchie definiert ist.

Eine Tabelle mit alternativen Hierarchie muss so aufgebaut sein:

hierarchie_id	integer
key	char(10)
parent	char(10)

Zum Aufbau der Hierarchie werden weiterhin alt\_hier\_id,treeviewtable und treeviewid benutzt.

## 2.4 Spezielle Details

### 2.4.1 Felder auf der Maske verstecken

Wenn Felder auf der Maske versteckt werden sollen, gibt es zwei Möglichkeiten:

- Feldart 13 → das Feld ist versteckt und wird intern nicht aufgebaut, im Masken-XML ist es aber enthalten

- Feldart beliebig, Eintrag in Spalte attribut: hidden

Das Feld wird intern aufgebaut und kann auch im Masken-SQL per FreeMarker benutzt werden, es wird aber keine Auswahlmöglichkeit auf der Maske angezeigt (benutzt bisher für Feld Kostenstelle, das nicht angezeigt werden sollte, im Masken-SQL aber schon für Rechtekontrollen benutzt wird)

- Feldart 999 (ab SuperX3.5rc2): Feld wird gar nicht erst aus Datenbank eingelesen, also ob nicht existent

Bei Benutzung der erweiterten kamerale Rechte SxFinRechte:

Auch wenn auf der Maske nicht alle kamerale Felder benötigt werden (z.B. Titel) müssen diese als versteckte Felder vorhanden sein, damit Querabhängigkeiten in Maskenbuttons z.B. FB SxFinRechte(,..,“<<Titel>>“,....) aufgelöst werden können!

Bei sehr vielen versteckten Feldern rutscht der Abschicken-Button nach unten, da auch versteckte Felder (noch) für die absolute Positionierung berücksichtigt werden. Trick versteckte Felder in felderinfo auf y=-1 setzen, dann kommen sie nicht in Reihenzählung.

## 2.5 Abfragemakros (einschl. Schleifen u. Grafiken)

Makros sind Abfragen, die mehrere andere Abfragen hintereinander ablaufen lassen.

Makros haben die üblichen Einträge in maskeninfo, felderinfo, masken\_felder\_bez, mas-ke\_sachgeb\_bez etc.

Die Spalte macro in der maskeninfo hatte im vorherigen Jahrtausend mal was damit zu tun, ob eine Maske ein Makro ist oder nicht – jetzt steuern die Einträge in der Spalte nur, ob eine Maske nur im Applet, nur im XML-Frontend oder in beiden erscheinen soll.

Der Eintrag sollte so sein, dass das Makro nur im XML-Frontend erscheint, da nur das XML-Frontend Makros unterstützt.

Welche Einzelabfragen ein Makro ausführen soll, wird in macro\_masken\_bez eingetragen.

maskeninfo_id1	maskeninfo-tid des makros,
maskeninfo_id2	maskeninfo-tid der Einzelabfrage,
active	0/1 deaktivieren möglich
sortnr	zur Reihenfolgebestimmung der Durchführung

also z.B.

maskeninfo_id1	maskeninfo_id2
12000	10000
12000	10050

Das Makro 12000 führt die Einzelabfragen 10000 und 10050 aus.

Das „select\_stmt“ eines Makros in der `maskeninfo` wird nicht benutzt, wichtig sind hingegen die Felder, z.B. `Lehreinheit` und `Semester`.

Diese Felder werden zur Auswahl angeboten, wenn das Makro im XML-Frontend angeklickt wird. Klickt man auf Abschicken werden die einzelnen Unterabfragen durchgeführt, wobei das ausgewählte Semester und die ausgewählte Lehreinheit auch automatisch in jeder Unterabfrage ausgewählt werden (Die Feldnamen müssen aber gleich sein, heißt das Feld in der Makromaske `Lehreinheit`, in der Einzelabfrage aber `Org.Einheit`, kann keine automatische Zuordnung erfolgen. Die Felder müssen gleich heißen oder man muss auf der Makromaske das Feld `Org.Einheit` zusätzlich aufnehmen).

### 2.5.1 Makros und Sichten

Wenn in den Submasken eines Makros verschiedene Sichten benutzt werden, müssen auf der Hauptmaske alle in den Untermasken benötigten Sichten auswählbar sein.

Sonst gelten ggfs. vorgenommene Standänderungen nur für die Sichten der Hauptmaske nicht für die der Untermasken.

Alternativ müsste man prüfen, ob Sichten von der Hauptmaske (Organigramm-Sicht) ihren Stand auch auf alle Sichten in den Untermasken übertragen, die von der gleichen Art sind (wahrscheinlich eher gefährlich).

### 2.5.2 Makro-Schachtelung

Makros sollen geschachtelt werden können, ein Makro soll also Untermakros aufrufen können.

Für ein Statistikheft könnte man dann einzelne Makros zu den Abschnitten des Hefts erstellen und testen und später dann ein „Mastermakro“ für das ganze Heft.

Auf Datenbankseite könnte man das recht einfach ermöglichen, wenn auch in `maskeninfo_id2` können Makros angegeben sein können.

z.B. Lehrbericht 10600,  
Untermakro Einschreibungen 10800,  
Einzelne Maske 10060, 10050

#### **makro\_maske\_bez**

maskeninfo_id1	maskeninfo_id2	sort
10600	10800	1
106005	10050	2
10800	10060	1
10800	10070	2

## 2.5.3 Schleifenfunktion

### 2.5.3.1 Grundlagen

Ein Makro soll automatisch mehrmals durchgeführt werden, z.B. für alle Lehreinheiten.

Erweiterung von **makro\_maske\_bez**

#### makro\_maske\_bez

Feld	Beispiel	Kommentar
masken-info_id1	10800	
masken-info_id2	10050	
active	0/1	0 zum zeitweisen deaktivieren
sort	1	
schleifenrelation	<<SQL>> select key_apnr,name from organigramm where lehre=1 and orgstruktur=30 order by 2	
schleifenfeldname	Org.Einheit	
schleifenstand	1.1.2005 oder <<Org. Einheit-Stand>> oder <<today>>	falls das schleifenfeld einen Stand benötigt
schleifensicht	13 oder <<Org. Einheit-Sicht>>	falls das schleifenfeld Sichten unterstützt (art=12) und mehr als eine Sicht zur Auswahl steht, muss die gewünschte Sicht (tid) angegeben werden
aktion		

Im Makro 10800 wird die Maske Stud. Allg (10050) aufgerufen.

Da Feld *schleifenrelation* gefüllt ist, wird <<SQL>> ausgeführt und Stud. Allg. mehrmals entsprechend der Anzahl gefundener Einträge ausgeführt, dabei wird immer ein Eintrag das Feld mit dem *schleifenfeldnamen* „Org. Einheit“ eingesetzt.

### 2.5.3.2 Schleife über ein anderes Makro

Um nicht eine einzelne Maske, sondern einen Makro mit mehreren Abfragen mehrmals per Schleife aufzurufen, muss ein übergeordnetes Makro geschaffen werden.

Im folgenden Beispiel gibt es ein Makro, dass Studierenden und Absolventenzahlen ausgibt (15000). Dies wird vom übergeordneten Makro 16000 für alle Lehreinheiten aufgerufen.



maskeninfo_id1	maskeninfo_id2	sort	schleifenrelation	schleifenfeldname
16000	15000	1	<<SQL>> select key_apnr,name from organi- gramm where lehre=1 and orgstruktur=30 order by 2	Org.Einheit
15000	10050	1		
15000	11930	2		

### 2.5.4 Spezielle Auswahlwerte hinterlegen

Beim Durchführen von Makros soll es möglich sein, für einzelne Felder einzelner Masken spezielle Auswahlwerte zu hinterlegen.

z.B. bei einem Makro mit 10 Studierendenabfragen, die standardmäßig die Haupthörerzahl ausgeben, soll bei einer Maske stattdessen beim Feld Hörerstatus Nebenhörer ausgewählt sein.

Lehrbericht (10600) ist ein Makro

makro maske sort

```
10600      10070  1
           10050  2
           10090  3
           10100  4
           10050  5
           11500  6
           10050  7   (hier Nebenhörer gewünscht)
           10050  8
```

...

Felderinfo der „MakroMaske“ (10600) definiert Felder, die sich auf alle Untermasken beziehen

bei einer einzelnen Untermaske (z.B. Stud. Allg. 10050) werden Felder gefüllt

**NEU:**

**Ein solcher Eintrag überlagert ggfs. in der Makro-Maske vorkommende Felder gleichen Namens!**

**makro\_feld\_wert**

Makro	Sortnr	feldername	value(char)	sicht	stand
10600	7	Hörerstatus	'N' für Nebenhörer	bei Sicht- feldern z.B. 13 oder <<Org. Ein- heit-Sicht>>	z.B. 1.1.2005 oder today oder <<Org. Einheit- Stand>>

Die Werte müssen genauso so eingetragen werden, wie sie vom SQL in der relation-Spalte des zugehörigen Felds geliefert werden.

z.B. steht für das Feld Staatsangehörigkeit in relation:

```
<<SQL>> select tid,text from aggreg_bland where tid in (0,1) order by text;
```

Ergebnis

```
tid  text
```

```
1   Ausland
```

```
0   Deutschland
```

Wenn Ausland vorbelegt werden soll, muss in makro\_feld\_wert als value nur

1

eingetragen werden.

**TODO**

Man sollte auch eine gewünschte Sicht mit einem gewünschten Stand hinterlegen können.

**2.5.5 Zukünftig: Feldnamen-Synonyme**

Wenn besonderes Problem tritt auf, wenn inhaltlich gleiche Felder in unterschiedlichen Masken unterschiedlich heißen, z.B. Institution, Lehreinheit, Org. Einheit oder Semester, Seit Semester, Start-Semester

Auf der Makro-Maske sollte idealerweise nur ein Feld stehen, z.B. Org. Einheit.

(Wenn verschiedene Sichten benutzt werden, müssen alle in den Untermasken benötigten Sichten auswählbar sein, s.o.).

Man müsste irgendwo Synonyme hinterlegen können.

z.B. makro\_feld\_alias

makro	integer	15000	
-------	---------	-------	--

feldname	char	Org. Einheit	
alias	char	Lehreinheit	
sortnr	integer	null für grundsätzlich oder sonst sortnr in einem Makro heißt, nur für die Abfrage x and sortnr=3	

## 2.5.6 Aktionen (Grafikerzeugung)

### 2.5.6.1 Grundlagen

In der Tabelle `macro_masken_bez` gibt es eine Spalte `aktion`, in der verschiedene Aktionen definiert werden können.

Derzeit sind folgende Aktionen möglich

Name	Beschreibung
<code>showTable</code>	Die Ergebnistabelle anzeigen. Diese Aktion wird standardmäßig durchgeführt, wenn bei <code>aktion</code> nichts angegeben ist.
<code>createChart-x</code>	Eine Grafik erzeugen – s. Kapitel Grafikerzeugung.

Mehrere Befehle sind durch `|` zu trennen, die Reihenfolge spielt eine Rolle.

`showTable|createChart-1` zeigt erst die Tabelle und danach die Grafik

`createChart-1|showTable` zeigt erst die Grafik und danach die Tabelle

Sofern die Spalte `aktion` null oder einen Leerstring enthält, wird die Aktion `showTable` durchgeführt.

In den Beispielen wird nach Java Konvention das erste Wort klein und danach jeder Wortanfang groß geschrieben, Sie können aber auch alles klein oder alles groß schreiben.

### 2.5.6.2 Grafikerstellung

#### 2.5.6.2.1 Grundlagen

Als Erstes muss irgendwo hinterlegt werden, dass nachdem eine Abfrage durchgeführt wurde auch eine Grafik erzeugt werden soll.

Dies geschieht im Feld `aktion` der Tabelle `makro_masken_bez`.

Das Kommando lautet:

`createChart-graphicformatTid`

z.B.

**makro\_maske\_bez**

maskeninfo_id1	maskeninfo_id2	sort	schleifenrelation	Schleifenfeldname	aktion
10800	10050	2			createChart- 2

In der Tabelle graphicformat werden Attribute für eine Grafik hinterlegt, die graphicformatTid im Kommando verweist auf einen Eintrag

**Tabelle graphicformat**

tid	
charttype	bar (Säulen), hbar (Balken), pie (Torten), line (Linien)
caption	Studierendenzahl (Personen) nach Semestern
width	400
height	400
caption x	Semester
caption y	Studierende
line x (Trennlinien)	0 oder 1
line y	0 oder 1
showvalues (Wert bei Balken anzeigen)	0 oder 1
moreattribs	

Da es ja noch eine ganze Menge weiterer Attribute, z.B. Farben von Balken 1-15, Intervalle auf der Y-Achse (1000,2000,3000 oder 1500, 3000, 5000) etc geben kann, haben wir überlegt, nicht für jedes potentielle Attribut eine eigene Spalte in der Tabelle anzulegen (irgendwas fehlt später stimmt auch noch), sondern ein BLOB-Feld moreattribs, in dem weitere Attribute nach der Syntax

attributname=wert| attributname =wert  
eingetragen werden können

z.B.

color1=100,200,0|maxWertYAchse=18000

Wenn es in Ergebnistabellen Leerzeilen gibt, werden diese für die Grafikerstellung automatisch ignoriert.

### 2.5.6.2.2 MoreAttribs

Die Attribute sind nicht case-sensitive. Statt nach Java Convention ignoreRowsWith: kann man auch ignorerowswith oder IGNOREROWSWITH schreiben.

Die Reihenfolge der Attribute spielt keine Rolle.

Attribute, die bei allen Grafiken anwendbar sind

Name	Beispiel / Erläuterung
ignoreRowsWith:	ignoreRowsWith:Summe Alle Zeilen der Ergebnistabelle, bei denen in der ersten Spalte das Stichwort „Summe“ vorkommt, werden bei der Grafikerstellung ignoriert.
ignoreColNo:	ignoreColNo:4 Spalte vier wird nicht ausgegeben
colorX:	color1:redcolor2:155,0,30color3:gray30% Definition von Farben über Systemname, mit Komma getrennten RGB-Werten oder Grauabstufung von 0% weiß bis 100% schwarz
flipFlop	Ganz zum Schluß werden Zeilen und Spalten ausgetauscht. Einträge für ignoreRows und ignoreColNo beziehen sich auf die ursprüngliche Tabelle

### 2.5.6.2.3 Säulendiagramme

charttype: bar oder bar3D

moreAttribs

useOnlyColNo:	useOnlyColNo:2 wenn nur 2. Spalte ausgegeben werden soll
---------------	---

### 2.5.6.2.4 Balkendiagramme

charttype hbar oder hbar3D (für horizontal bar)

MoreAttribs

useOnlyColNo:	useOnlyColNo:2 wenn nur 2. Spalte ausgegeben werden soll
---------------	---

### 2.5.6.2.5 Tortendiagramme

Bei einem Tortendiagramm wird üblicherweise die erste und zweite Spalte visualisiert.  
Im Beispiel die Hauptnutzfläche aufgeteilt nach Gebäuden.

Ge- bäude	Hauptnutz- fläche (HNF)	Nebennutz- fläche (NNF)	Verkehrs- fläche	Funktions- fläche	Sonstige Fläche	Gesamt- fläche
GA	31,56	14,00	0,00	0,00	0,00	31,56
GABF	83,50	15,00	0,00	0,00	0,00	83,50
GB	810,20	123,00	0,00	0,00	0,00	810,20
GBCF	43,53	20,00	0,00	0,00	0,00	43,53
UB – Bibliothek	944,29	183,00	0,00	0,00	0,00	944,29

Falls nicht die Werte der zweiten Spalten visualisiert werden sollen, kann in der Tabelle `graphicformars` unter `moreAttribs` der Parameter

`useNo:` angegeben werden, also z.B. `useNo:3` für eine Darstellung der Nebennutzfläche.

Sie können bei Tortendiagrammen auch das Attribut `flipFlop` angeben.

Dann wird standardmäßig die zweite Zeile (hier das Gebäude GA) mit HNF,NNF,Verkehrsfläche etc als Torte dargestellt.

Um die Gesamtfläche auszublenden fügen Sie noch `ignoreColNo:7` hinzu, das funktioniert da der Austausch von Zeilen und Spalten (`flipFlop`) ja grundsätzlich als Letztes passiert.

Um statt der zweiten Zeile (GA), die dritte Zeile (GABF) zu visualisieren müssten Sie `useNo:3` angeben.

### 2.5.6.3 spezielle Stylesheets benutzen

Zukünftig:

Für einzelne Masken innerhalb eines Makros können auch Stylesheets hinterlegt werden, die angewendet werden sollen

aktion

stylesheet-1;stylesheet-2

Nummer ist tid in `sx_stylesheets`

## 2.6 Dokumentation von Abfragen

### 2.6.1 Glossare

Die Statistiken in SuperX ist nicht immer für Außenstehende "selbsterklärend", und insbesondere bei Kennzahlen und kondensierten Werten sollten die Konzepte mit einem Glossar versehen sein.

Die Frontends von SuperX bieten drei Möglichkeiten der Dokumentation:

- Dialogelemente auf den Masken können mit einem "Tool-Tip" versehen werden, d.h. bei Mausbewegung über den Button wird eine Erläuterung angezeigt.
- Ergebnistabellen können mit einem Glossar versehen werden, das die in der Tabelle benutzten Begriffe auf einer zweiten Seite erläutert.
- Umfangreichere Hilfetexte sind über die kontextabhängigen Hilfetexte zu einer Maske und Ergebnistabelle verlinkt. Dies ist in der [Entwickleranleitung](#) (S. 62) dokumentiert.

Für die ersten beiden Dokumentationsarten wird in SuperX die Tabelle `sx_captions` gepflegt, die Felderläuterungen und allgemeine Schlüsselörter dokumentieren. Die Dokumentation ist sogar mehrsprachig möglich.

#### 2.6.1.1 Allgemeine Schlüsselwörter

Allgemeine Schlüsselwörter sind der Tabelle `sx_captions` definiert, man erkennt sie, daran dass die Spalte `id` gefüllt ist (`table_name`, `field_name` und `record_no` hingegen leer)

tid	id	table_name	field_name	record_no	locale	contents_short	contents_long	sachgebiete_id
1	studiengang				de	Studiengang	Studiengänge definieren sich durch das Fach, die Vertiefungsrichtung, durch Haupt- oder Nebenfach sowie den Abschluss.	16
2	studiengang				en	Subject / Degree	A combination of subject and degree as well as the major-minor distinction	16
3	stud_general				de	Studierende allgemein		
4	stud_general				en	students (general)		

Im Beispiel wird der Tag **studiengang** definiert.

Dieser Tag wird an beliebiger Stelle (Maskennamen, Überschriften, select\_stmt, XIL-Proplist, XSL-Dateien, etc) durch den Eintrag `contents_short` der aktuellen Locale ersetzt.

### 2.6.1.2 Der Spezialfall Maskenfelder

Für die Erläuterung von Maskenfeldern können kurze und längere Hilfetexte hinterlegt werden. Die kurzen Texte dienen als Beschriftung des Feldes (überschreiben als den "Feldnamen"), und die langen Texte erscheinen als Tool-Tip bei Mausbewegung auf den Button. Im Ausdruck werden die Maskenfelder wahlweise auf einer separaten Seite dokumentiert.

Damit nicht für jedes einzelne Maskenfeld ein Eintrag gemacht werden muss, kann ein Hilfetext über seinen Namen auch mehreren Maskenfeldern zugeordnet werden; in diesem Fall ist die Spalte `record_no` leer.

Für Felder aus der Tabelle `felderinfo` schaut SuperX nach, ob in der Tabelle `sx_captions` ein Eintrag für die Tabelle `felderinfo`, `field_name` `studiengang` und `record_no` = 10050 oder null vorhanden ist

Im folgenden Beispiel ist ein Maskenbutton "Studiengang" erläutert, der in dieser Weise und bei dem Feld Nummer 10050 dokumentiert sein soll.

tid	id	table_name	field_name	record_no	locale	contents_short	contents_long	sachgebiete_id
9		felderinfo	studiengang	10050	de	Grundständiger Studiengang	Ein Studiengang im grundständigen Studium	16
10		felderinfo	studiengang	10050	en	Degree program		16

Wenn Sie den Erläuterungstext bei allen Feldern mit dem Namen "studiengang" erscheinen lassen wollen, dann müssen Sie das Feld `record_no` leer lassen.

### 2.6.1.3 Änderung von Glossaren im XML-Frontend

Im XML-Frontend gibt es komfortable Möglichkeiten zur Änderung von Glossaren. Melden Sie sich als Administrator im XML-Frontend an. Zunächst zeigen wir, wie allgemeine Erläuterungen zur Makse sowie Erläuterungen von Ergebnisspalten erzeugt werden, dann werden Felderläuterungen eingepflegt.

#### 2.6.1.3.1 Maskenerläuterung

Masken bearbeiten wir im XML-Frontend im Menü "Maske Suchen", wie es im [Tutorial](#) (S. 7) beschrieben ist.



Wir wählen bei Maske suchen eine Maske, die wir dokumentieren wollen, z.B. die Abfrage **Studierende und Studienanfänger nach Geschlecht**.

12.05.2005 hilfe | über

### Maske suchen

Bitte schränken sie Ihre Auswahl ein:

Sachgebiet

Maske

Titelstichwort

Wir schicken das Formular ab, und erhalten Bearbeitungsmöglichkeiten zur Maske. Wir wählen hier "Bearbeiten".

### Maske suchen

Maske: 16020 - Studierende und Studienanfänger nach Gesc... ; Stand: 01.01.2003

Maske Nr	Name	Sachgebiet	Bearbeiten	Sachgebiete	Stylesheets
16.020	Studierende und Studienanfänger nach Geschlecht	Studierende			

Datensatz 1 - 1 von insgesamt 1 Satz.

Wir erhalten das Beabrietungsformular der Maske. Diesmal kümmern wir uns nicht um das Feld "select\_stmt", alsod er Script, sondern um die Dokumentation.

Su  
Sug

### Maskeninfo verwalten

In diesem Formular können Sie Masken verwalte

Tid:

Name:

Select\_stmt: 

```
--Wenn Freemarker eingesetzt wird, muss der folgende Kommentar (case insensitive)
irgendwo in der Abfrage stehen
--Freemarker Template
<#include "SQL_lingua_franca"/>
<#include "SuperX_general"/>

--start Datentabelle
<@selectintotmp
select="L.text, L.lehr, L.stg as ch30_fach, L.vertfg as ch39_vertief, L.kz_fach,
fach_nr,L.abschluss as ch35_ang_abschluss,summe, ca12_staat, geschlecht,
fach_sem_zahl, kz_rueck_beur_ein"
```

Xil\_proplist: 

```
Column CID=3 heading_text=" " center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=1 explanation=""
Column CID=4 heading_text="1. FS\n gesamt" center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=10 explanation="@@@1Fachsemester@@@"
Column CID=5 heading_text="1. FS\n in %" center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=10 explanation="wie vorherige Spalte in %"
Column CID=6 heading_text="1. HS\n gesamt" center_heading
row_selectable col_selectable rightJust heading_platform readonly
```

Weiter unten auf dem Formular stehen die Spaltenbeschriftungen, bei Spalte 5 z.B. "wie vorherige Spalte". Zunächst schauen wir uns die Erläuterung der Maske an (Feld "Erläuterung").

Xil_proplist	Column CID=3 heading_text=" " center_heading row_selectable col_selectable rightJust heading_platform readonly width=1 explanation="" Column CID=4 heading_text="1. FS\n gesamt" center_heading row_selectable col_selectable rightJust heading_platform readonly width=10 explanation="@@@1Fachsemester@@@" Column CID=5 heading_text="1. FS\n in %" center_heading row_selectable col_selectable rightJust heading_platform readonly width=10 explanation="wie vorherige Spalte in %" Column CID=6 heading_text="1. HS\n gesamt" center_heading row_selectable col_selectable rightJust heading_platform readonly	
Chart_xtitel	Studienfach	
Chart_ytitel	Anzahl bzw. Anteil	
Erläuterung	<html>Gesamtzahl rückgemeldeter Studieren	
Cleanup_stmt	drop table tmp_rs_final2;	
Default_file	studallg.dat	
Spezielles Frontend	<input type="button" value="v"/>	Applet=0, XML=1, beide=2
Breite	850	
Höhe	600	
Ampel	0	
Hilfe	1	
Hinweis	<<SQL>> select erlaeuterung from koepfe_oder_faelle where apnr = '<<Köpfe oder Fälle ?>>';	

Im Applet wird der Erläuterungstext bei Mausklick auf den Button "Erläuterung" angezeigt:

Themenauswahl | Maske | Tabelle

**Studierende und Studienanfänger nach Geschlecht**

Parameter:  
Köpfe oder Fälle ? = Köpfe; Semester = SS 2005; Fächer = keine Einschränkung (Fächer (Intern)) - Stand 12.05.2005; Hörerstatus = alle; Aggregation Fach = Fächer + Studiengänge; Status = Alle ohne Beurl.; User=superc;

Stand: 01.01.2003

Ebene	Studiengang	Gesamt-zahl	1. FS gesamt	1. FS in %	1. HS gesamt	1. HS in %
Summe Fach (intern)	Fach (intern)	1.416	13	0,92		
Fach (intern)	Allgemeine Sprachwissenschaft	1				
Studiengang	Allg. Sprachwiss. Allg. Spr./Grammatik/Sem. Grun...	1				
Fach (intern)	Bildungsplanung/Instructional Design	6				
Studiengang	Bildungspl./Instr. Design Bakkalaureus Artium H...	1				
Studiengang	Bildungspl./Instr. Design Bakkalaureus Artium H...	5				
Fach (intern)	Biologie	62				
Studiengang	Biologie Diplom Prüf.-Ordn. 2000	10				
Studiengang	Biologie Diplom Prüf.-Ordn. 2002	23				

234 Sätze gefunden

Erläuterung

Wie sehen den Text aus der Maskenerläuterung sowie für jede dokumentierte Spalte den Text der Spaltenerläuterung.

Erläuterung zu Studierende und Studienanfä...

**Erläuterung zur Abfrage Studierende und Studienanfänger nach Geschlecht**

Gesamtzahl rückgemeldeter Studierender in **einem** Semester:  
Nach Studiengang, Geschlecht, 1.FS, 1.HS

**Spalte 2** Stg

**Spalte 5** Die Anzahl der Studierenden im ersten Fachsemester

**Spalte 6** wie vorherige Spalte in %

Drucken

Der Erläuterungstext von Spalte 5 ist ein Sonderfall, hier wurde oben in der Maske mit einem Platzhalter namens "@@@1Fachsemester@@" gearbeitet. Der Inhalt für den Platzhalter wird in der Beschriftungstabelle gepflegt. Dazu müssen wir in das Menü "Administration -> Beschriftung suchen" gehen.

Wir wählen z.B. das Stichwort "Fachsem" aus, und schicken das Formular ab.

### Beschriftungen suchen

Bitte schränken sie Ihre Auswahl ein:

id

Stichwort (kurz)

Stichwort (lang)

Sprache

Tabelle

Feldname

Maske

Es erscheinen als Ergebnis mehrerer Zeilen, die erste Zeile enthält unseren Platzhalter (im Feld "id"). Wir bearbeiten diesen Eintrag.

Export: [Druckversion](#) [XML](#) [Text](#) [RTF](#)

### Beschriftungen suchen

Stichwort (kurz): **Fachsem** ; Sprache: **Deutsch** ; Stand: 01.01.2003

id	Tabelle	Feld	Datensatz Nr.	Sprache (kurz)	Inhalt (kurz)	Inhalt (lang)	Bearbeiten
1Fachsemester				de	1. Fachsemester	Die Anzahl der Studierenden...	
	felderinfo	bis Fachsemester		de	bis Fachsemester		
	felderinfo	ab Fachsemester		de	ab Fachsemester		

Datensatz 1 - 3 von insgesamt 3 Sätzen.

Der Erläuterungstext zur Spalte steht im Feld contents\_long. Genau dieser Text wurde oben in der Erläuterung der Spalte 5 angezeigt. Sie speichern die Angaben mit "Speichern".

**Beschriftungen** In diesem Formular können Sie Beschriftungen bearbeiten

tid|223

Id|1Fachsemester

Table\_name|

Field\_name|

Record\_no|[NULL]

Locale|Deutsch

Contents\_short|1. Fachsemester

Contents\_long|Die Anzahl der Studierenden im ersten Fachsemester

Equalitystatus|0

Sachgebiete\_id|Studierende

### 2.6.1.3.2 Feldbeschriftungen ändern

Für die Erläuterung von Maskenfeldern können kurze und längere Hilfetexte hinterlegt werden. Die kurzen Texte dienen als Beschriftung des Feldes (überschreiben als den "Feldnamen"), und die langen Texte erscheinen als Tool-Tip bei Mausbewegung auf den Button. Im Ausdruck werden die Maskenfelder wahlweise auf einer separaten Seite dokumentiert.

Damit nicht für jedes einzelne Maskenfeld ein Eintrag gemacht werden muss, kann ein Hilfetext über seinen Namen auch mehreren Maskenfeldern zugeordnet werden; in diesem Fall ist die Spalte `record_no` leer.

Für Felder aus der Tabelle `felderinfo` schaut SuperX nach, ob in der Tabelle `sx_captions` ein Eintrag für die Tabelle `felderinfo`, `field_name` `studiengang` und `record_no` = 10050 oder null vorhanden ist

Im folgenden Beispiel ist ein Maskenbutton "Studiengang" erläutert, der in dieser Weise und bei dem Feld Nummer 10050 dokumentiert sein soll.

tid	id	table_name	field_name	record_no	locale	contents_short	contents_long	sachgebiete_id
9		felderinfo	studiengang	10050	de	Grundständiger Studiengang	Ein Studiengang im grundständigen Studium	16
10		felderinfo	studiengang	10050	en	Degree program		16

Wenn Sie den Erläuterungstext bei allen Feldern mit dem Namen "studiengang" erscheinen lassen wollen, dann müssen Sie das Feld `record_no` leer lassen.

Im folgenden Beispiel wollen wir die Beschriftungen der Maskenfelder bearbeiten. Gehen Sie im Menübaum zum Eintrag

Administration -> **Beschriftungen suchen.**

Wir wählen z.B. das Stichwort "Staats" aus, und schicken das Formular ab.

**Beschriftungen suchen**

Bitte schränken sie Ihre Auswahl ein:

id

Stichwort (kurz)

Stichwort (lang)

Sprache

Tabelle

Feldname

Maske

Es erscheint als Ergebnis ein Felderinfo-Eintrag zu diesem Thema. Wir sehen, dass nur die Kurzbeschriftung gefüllt ist. Wenn wir das Konzept mit einem Tooltip versehen wollen, gehen wir rechts auf "Bearbeiten"

Export: [Druckversion](#) [XML](#) [Text](#) [RTF](#)

### Beschriftungen suchen

Stichwort (kurz): **Staats** ; Sprache: **Deutsch** ; Stand: 01.01.2003

id	Tabelle	Feld	Datensatz Nr.	Sprache (kurz)	Inhalt (kurz)	Inhalt (lang)	Bearbeiten
	felderinfo	Staatsangehörigkeit		de	Staatsangehörigkeit		

Datensatz 1 - 1 von insgesamt 1 Satz.

Wir können eine Langbeschreibung im Feld "contents\_long" einfügen. Der dortige Text wird als Tooltip angezeigt. Sie speichern die Angaben mit "Speichern".

**Beschriftungen** In diesem Formular können Sie Beschriftungen bearbeiten

tid|215

Id|leer-

Table\_name|felderinfo

Field\_name|Staatsangehörigkeit

Record\_no|[NULL]

Locale|Deutsch

Contents\_short|Staatsangehörigkeit

Contents\_long|Ausländische Studierende sind weibliche oder männliche Personen mit einer anderen als der deutschen Staatsangehörigkeit, die an einer deutschen Hochschule immatrikuliert sind.

Equalitystatus|[NULL]

Sachgebiete\_id|Studierende

[Speichern](#) [<< Erster](#) [< Vorheriger](#) [Nächster >](#) [Letzter >>](#) [Löschen](#) [Neu](#)

Im Applet wird derText dann bei Mausbewegung über den Button **Staatsangehörigkeit** wie folgt angezeigt:

Themenauswahl **Maske** | Tabelle

**Studierende und Studienanfänger nach Geschlecht**

Köpfe oder Fälle ? Köpfe Semester SS 2005 Stichtag

Fächer

Abschluß

bis Fachsemester Hörerstatus alle

Staatsangehörigkeit Hochschulzugangsberechtigung

Ausländische Studierende sind weibliche oder männliche Personen mit einer anderen als der deutschen Staatsangehörigkeit, die an einer deutschen Hochschule immatrikuliert sind.

Aggregation Fach Fächer + Studiengänge Status Alle ohne Beur.

## 2.6.2 Erzeugung der SuperX-Hilfe im Javahelp-Format

Die SuperX-Hilfe im Applet besteht aus einem Archiv im Javahelp-Format. Die Hilfetexte sind in den Modulen erzeugt und können problemlos integriert werden. Falls Sie eigene Hilfetexte einbinden wollen, müssen Sie wie folgt vorgehen:

1. Erzeugen Sie html-Seiten mit der Hilfe (html 3.2)

## 2. Binden Sie die Dateien in die Mapping-Datei ein

(`$SUPERX_DIR/webserver/tomcat/webapps/superx/applet/javahelp/map.jhm`)

## 3. Falls die Hilfeseiten kontextabhängig abrufbar sein sollen, müssen die Titel der Mapping-Einträge folgenden Konventionen folgen:

- Allgemeine Beschreibungen der Abfragen lauten A<<TID>>.htm
- Beschreibungen der Masken lauten M<<TID>>.htm
- Beschreibungen der Ergebnistabellen lauten T<<TID>>.htm

Am Anfang ist es hilfreich, die vorhandenen Hilfetexte als Vorlage zu benutzen.

Die Javahilfe kann auch komfortabler mit dem Memtext-Autorensystem aus einer Word-Datei erzeugt werden. Details dazu siehe <http://studio.memtext.de>.

## 2.7 Werkzeuge zur Masken-Entwicklung

### 2.7.1 Shell-Scripte

#### 2.7.1.1 Masken-Verwaltung

Zum Erzeugen und Verändern von Masken gibt es unter UNIX eine Kommandoschnittstelle, die auf dem Gebrauch folgender Skripte beruht. Die Skripte stehen unter dem Verzeichnis

`$SUPERX_DIR/db/masken`

und erzeugen oder verwenden Dateien in dem gegenwärtigen Arbeitsverzeichnis. Nach dem Einspielen der Datenbank sollten Sie darauf achten, den Dateien Ausführungsberechtigung (`chmod 750 sx_*`) zu geben.

##### 2.7.1.1.1 Eine Maske suchen

Wenn Sie eine Maske suchen, sollten die die Felder `tid` oder `name` in der Tabelle `maskeninfo` durchsuchen. Das folgende Script macht dies automatisch:

#### **sx\_search\_mask**

Aufruf:	<code>sx_search_mask &lt;String&gt;</code>
Aktion:	<code>sx_search_mask</code> sucht die Masken, deren Name <code>&lt;String&gt;</code> enthält
Ausgabe: .	<code>tid, name</code> der gefundenen Masken

##### 2.7.1.1.2 Eine Maske sichern und entladen

Um eine Maske zu sichern, müssen Sie die entsprechenden Einträge in den Tabellen

1. `felderinfo`,
2. `masken_felder_bez`,
3. `maskeninfo`,

4. sachgeb\_maske\_bez,  
 5. maske\_system\_bez  
 selektieren und sichern. Für dies gibt es das Script `sx_select_mask`.

#### **sx\_select\_mask**

<b>Aufruf:</b>	<code>sx_select_mask &lt;TID&gt;</code>
<b>Aktion:</b>	<code>sx_select_mask</code> entlädt alle Metadaten aus den Tabellen <code>maskeninfo</code> , <code>felderinfo</code> , <code>masken_felder_bez</code> , <code>sachgeb_maske_bez</code> , <code>maske_system_bez</code> zur Maske mit <code>tid = &lt;TID&gt;</code> .
<b>Ausgabe:</b>	Fünf Dateien: 1. <code>&lt;TID&gt;_felderinfo.unl</code> , 2. <code>&lt;TID&gt;_masken_felder_bez.unl</code> , 3. <code>&lt;TID&gt;_maskeninfo.unl</code> , 4. <code>&lt;TID&gt;_sachgeb_maske_bez.unl</code> , 5. <code>&lt;TID&gt;_maske_system_bez.unl</code>

#### **2.7.1.1.3 Eine Maske neu einfügen**

Um eine Maske neu einzufügen, müssen Sie die entsprechenden Einträge in den Tabellen

6. `felderinfo`,  
 7. `masken_felder_bez`,  
 8. `maskeninfo`,  
 9. `sachgeb_maske_bez`,  
 10. `maske_system_bez`  
 einfügen. Dafür gibt es das Script `sx_insert_mask`.

#### **sx\_insert\_mask**

<b>Aufruf:</b>	<code>sx_insert_mask &lt;TID&gt; [&lt;neue TID&gt;] [j]</code>
<b>Aktion:</b>	<code>sx_insert_mask</code> lädt den Inhalt der fünf Dateien 1. <code>&lt;TID&gt;_felderinfo.unl</code> , 2. <code>&lt;TID&gt;_masken_felder_bez.unl</code> , 3. <code>&lt;TID&gt;_maskeninfo.unl</code> , 4. <code>&lt;TID&gt;_sachgeb_maske_bez.unl</code> , 5. <code>&lt;TID&gt;_maske_system_bez.unl</code> in die jeweiligen Tabellen der SuperX-Datenbank. Mit "j" wird die Sicherheitsabfrage umgangen.

Falls `<neue TID>` nicht angegeben wird, werden die Metadaten wieder mit der alten TID in die Datenbank eingespielt (=Update).

Falls `<neue TID>` angegeben wird, werden die Metadaten mit der neuen TID in die Datenbank eingespielt (=Insert). Dabei werden alle TIDs in den abhängigen Tabellen angepasst. So können Masken sehr einfach kopiert werden. Eine neue TID bekommt man durch die Wahl der nächsten Zehnerzahl, die grö-



ßer als die größte vorkommende Nummer ist. Die größte vorkommende Nummer erhält man durch Ausführung des folgenden SQL-Ausdrucks mit Hilfe des Kommandos SQL-Client:

```
select max(tid) from maskeninfo;
```

#### 2.7.1.1.4 Eine Maske löschen

Um eine Maske zu löschen, müssen Sie die Einträge in den oben genannten Tabellen entfernen. Dafür gibt es das Script `sx_delete_mask`

##### **sx\_delete\_mask**

Aufruf:	<code>sx_delete_mask &lt;TID&gt;</code>
Aktion:	<code>sx_delete_mask</code> löscht alle Metadaten aus den Tabellen <code>maskeninfo</code> , <code>felderinfo</code> , <code>masken_felder_bez</code> , <code>sachgeb_maske_bez</code> und <code>maske_system_bez</code> zur Maske mit <code>tid = &lt;TID&gt;</code> .

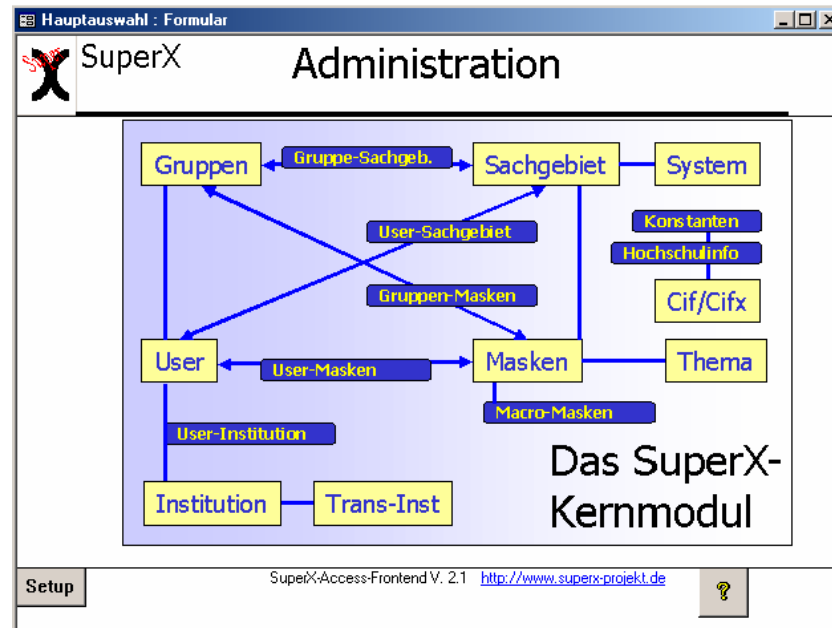
#### 2.7.1.2 Änderungen an einer Maske vornehmen

1. Selektieren der Metadaten der betreffenden Maske: `sx_select_mask <TID>`
2. Editieren der fünf Metadaten-Dateien „`<TID>_...`“
3. Abspeichern der neuen Metadaten: `sx_insert_mask <TID>`

### 2.7.2 Das Access-Frontend

Die Access-Datenbank enthält die Tabellen des Kernmoduls als Verknüpfungen und ermöglicht so ein leichtes Administrieren der Datenbank. Die Installation ist in der Installationsanleitung für [ODBC-Quellen](#) beschrieben. Die folgende Abbildung zeigt das Hauptmenü:

Das Frontend eignet sich zur Verwaltung von Usern, Gruppen, Sachgebieten und Masken sowie deren relationalen Verküpfungen (blaue Kästchen). Darüberhinaus sind Formulare für das Systeminfo, den Themenbaum und das Organigramm vorgesehen.






Probleme mit der Bedienung von Access gibt es immer dann, wenn Tabellen keine Primärschlüssel haben oder wenn die Felder mit den Primärschlüsseln nicht gefüllt sind. Mit der Version 2.1 erhalten alle Tabellen in SuperX (außer Datentabellen und Hilfstabellen, weil diese normal nicht manuell bearbeitet werden) Primärschlüssel. Wenn es dennoch Probleme gibt, empfehlen wir die Java-basierte [SQLWorkbench](#) (S. 68).

### 2.7.3 Maskenverwaltung mit MS Access

Das [Access-Frontend](#) ermöglicht die bequeme Änderung von Abfragen (für die Eingabe neuer Masken und Felder empfehlen wir eher die Abfragen im normalen Themenbaum). Nach dem Öffnen der Datei `/db/superx_frontend.mdb` können Sie unter *Masken* die einzelnen Masken von SuperX anwählen und öffnen. Sie erhalten im Formular `maskeninfo` ein Formular, das Eingaben oder Änderungen in der Tabelle `maskeninfo` ermöglicht.

Das Formular ermöglicht es, Masken zu ändern und zu erzeugen. Sie können eine TID vergeben und einen Namen eintragen.

Das `select_stmt` ist ein großes Textfeld und läßt sich besser durch Drücken der -Taste in einem separaten Fenster bearbeiten. Leider werden Tabulatoren im normalen Windows-Editor nicht korrekt dargestellt, deshalb befinden sich rechts noch zwei Buttons, mit denen Sie Masken in Word<sup>viii</sup> editieren können.

Mit dem Button  öffnen Sie das `select_stmt` in Word, und können dort Änderungen vornehmen. Mit dem Button  speichern Sie die Änderungen in der Datenbank, und Word wird geschlossen. Bitte beachten Sie, dass Sie die Dateien in Word nicht speichern müssen. Analog können Sie verfahren, wenn Sie das Feld `xil_proplist` bearbeiten. Um in Access sicherzustellen, dass Feldänderungen wirklich in der Datenbank gespeichert werden, sollten Sie sich einen Button zum Speichern von Datensätzen in die Access-Symbolleiste setzen (Extras -> Anpassen -> Befehle -> Datensatz speichern in eine häufig benutzte Symbolleiste ziehen).

Mit dem Button `Felderinfo` gelangen Sie zu den Feldern dieser Maske. Sie können die Felder dort bearbeiten. Beim Hinzufügen neuer Felder müssen Sie allerdings die jeweiligen `tids` manuell in die Tabelle `masken_felder_bez` eintragen.

Analog funktioniert die Bearbeitung der individuellen Stylesheets für eine Maske.

## 2.7.4 Weitere Tools

Durch die `odbc`- und `jdbc`-Treiber können beliebige Datenbankfrontends eingesetzt werden. Gute Erfahrungen gerade mit Tabellen ohne Primärschlüssel haben wir mit der `SQLWorkbench` von Thomas Kellerer gemacht. Exemplarisch für andere `jdbc`-Clients haben wir dieses Programm näher beschrieben.

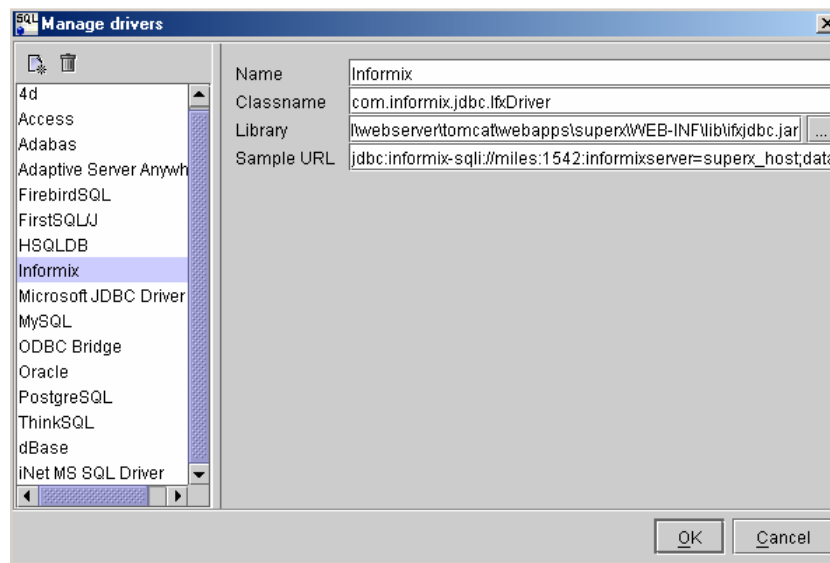
### 2.7.4.1 SQLWorkbench

Die [SQLWorkbench](#) arbeitet mit dem jdbc-Treiber jeweils von Postgres oder Informix. Sie ist Teil des SuperX-Clientpakets 3.0 oder höher, das Sie vom [Download-Bereich](#) des SuperX-Projektes laden können. In dem Clientpaket sind alle notwendigen Treiber und Profile bereits enthalten, deshalb empfehlen wir den Einsatz des Clientpakets.

Beim ersten Aufruf der Workbench können Sie Profile für Treiber und Datenbanken eingeben. Musterprofile für viele gängige Datenbanksysteme liegen vor. Leider ist der Informix-Treiber nicht dabei, deshalb muss dieser "von Hand" registriert werden. Gehen Sie dazu über File->Connect in das Feld "Manage Drivers". Dort können Sie einen Namen vergeben und die jdbc-Parameter übertragen. Die folgende Abbildung zeigt ein Beispiel:

Der Dialog zur Einrichtung von Datenbanktreibern am Beispiel Informix.

Die Parameter entsprechen denen, die Sie für das SuperX-Servlet in [db.properties](#) definieren. Der Informix-Treiber `ifxjdbc.jar` muss lokal gespeichert sein.

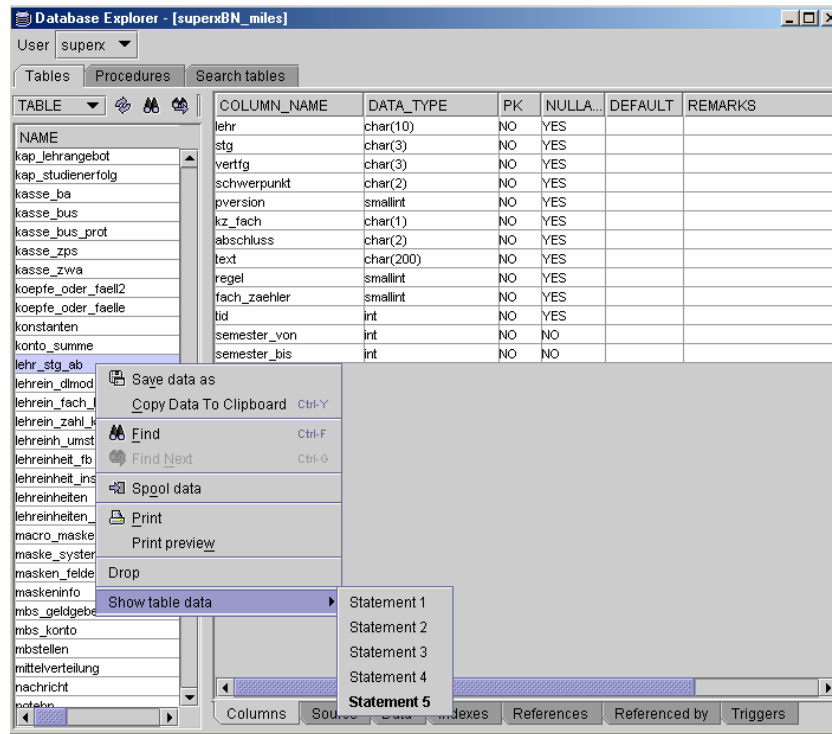


Im Dialog File -> Connect können Sie dann eine Datenquelle eintragen, und die Verbindungsparameter vervollständigen (Username, Passwort). Autocommit sollten Sie immer einschalten.

Interessant ist der Datenbank-Explorer (Tools -> Database Explorer), der es ermöglicht, die Datenbank nach Tabellen / Prozeduren etc. zu durchsuchen. Wenn eine Tabelle ausgewählt ist, kann sie auch über die Registerkarte "Data" editiert werden. Achten Sie darauf, dass Sie das Feld **Max. Rows** auf einen sinnvollen Wert setzen, z.B. 2000. Die SQLWorkbench ist gerade für die Arbeit mit Tabellen ohne Primärschlüssel geeignet, weil jede Änderung intern als Update formuliert wird. Der Nachteil ist, dass das Tool manchmal recht langsam ist, und dass nicht mehrere Zellen über Zwischenablage geändert / eingefügt werden können.

Sehr praktisch für die Entwicklung von SQL-Abfragen ist die Möglichkeit, zu jeder Tabelle eine select-String zu formulieren.

Markieren Sie die Tabelle im Database Explorer, und gehen Sie über das Kontextmenü auf Show table data, und wählen Sie ein Editorfenster aus. Der Select-String wird dann angezeigt.



Ein Nachteil bei Informix: Felder vom Typ `text` werden als binäre Arrays angezeigt und sind nicht editierbar. Z.B. die Felder `select_stmt` oder `xil_proplist` in der Tabelle `maskeninfo` müssen Sie mit einem anderen Tool, z.B. dem Access-Frontend über `odbc`, editieren.

Das Tool bietet außerdem eine Makrofunktion, und in neueren Versionen auch ETL-Funktionen über einen "Data Pumper", was es natürlich für SuperX besonders interessant macht. Weitere Tipps und Hilfen erhalten Sie im (gelungenen, aber englischen) Benutzerhandbuch.

## 2.7.5 Entwicklungsservlet für SuperX-Abfragen

Neu im Kernmodul3.5 ist ein spezielles Entwicklungsservlet, das die (verschiedenen Stadien der) Verarbeitung und Berichtserstellung in SuperX transparent macht.

Bisher wurde zu Entwicklungszwecken insbesondere das Applet oder die SQL-Angaben im SuperX-Manager genutzt, nun gibt es ein spezielles Entwicklungsservlet, das die Arbeit erleichtern soll.

### 2.7.5.1 Aufrufseite des Entwicklungsservlets

Es ist auf Ihrem Webserver unter der Adresse

<http://RECHNER:PORT/superx/servlet/de.superx.servlet.Entwicklung> erreichbar.

Wenn Sie eine kürze URL einrichten möchten, können Sie in der Datei

`webserver/tomcat/webapps/superx/WEB-INF/web.xml`

einen Eintrag hinzufügen:

```
<servlet>
    <servlet-name>
```

```

    Entwicklung
</servlet-name>
<servlet-class>
    de.superx.servlet.Entwicklung
</servlet-class>
</servlet>

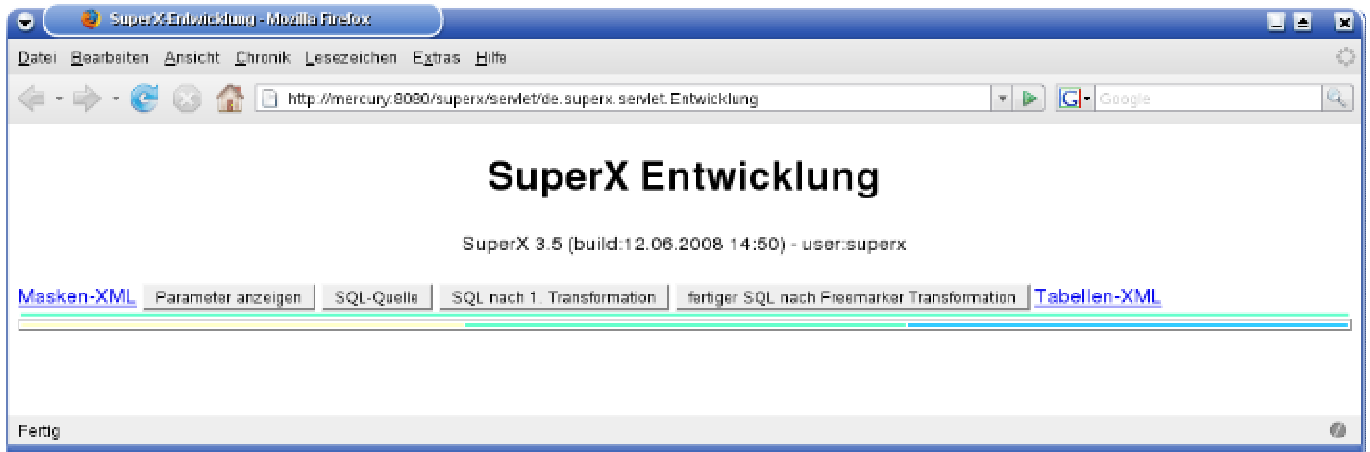
```

Dann können Sie nach einem Tomcatneustart, das Servlet auch unter der folgenden Adresse aufrufen:  
<http://RECHNER:PORT/superx/servlet/Entwicklung>

### 2.7.5.2 Funktionalität des Entwicklungsservlets

Mit dem Servlet kann man sich den Ausgangs-SQL, die Anpassungen durch die klassische SuperX-Transformation sowie den fertigen SQL nach FreeMarker-Transformation und auch die übergebenen Parameter ansehen. Außerdem kann man sich den Masken- und Tabellen-XML runterladen.

Hier eine Abbildung der Startseite:



Wenn Sie eine Abfrage ausführen, können Sie zunächst mit dem Link "Masken-XML" den XML-Code der Maske aufrufen (klicken Sie jeweils auf die Grafik, um sie zu vergrößern):

Hier die Maske:

17.06.2008

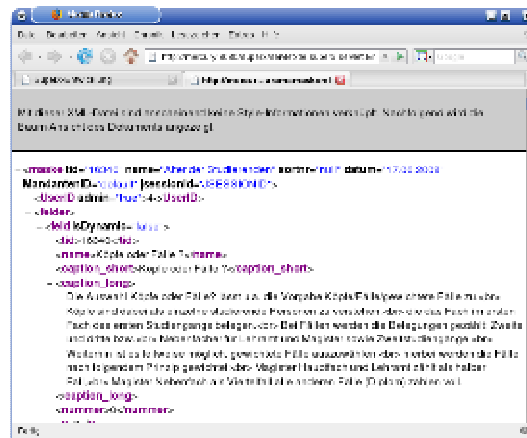
---

### Alter der Studierenden

Bitte schränken Sie Ihre Auswahl ein:

Köpfe oder Fälle? Köpfe	Sichtart Aktuelle Zahlen
Seit Semester 55.2007	bis Semester 55.2007
Fächer nichts gewählt	Status Alle ohne Best.
Hierarchiestatus alle	bis Fachsemester
Aggregation Fach Fächer + Studiengänge	Abschließen nichts gewählt
Geschlecht	Fiber Studienende

und hier der XML-Code:



Im nächsten Schritt können die vom Benutzer eingegebenen Parameter sichtbar gemacht werden:

SuperX Entwicklung

SuperX 3.5 (build:12.06.2008 14:50) - user:superx

Masken-XML | Parameter anzeigen | SQL-Quelle | SQL nach 1. Transformation

fertiger SQL nach Freemarker Transformation | Tabellen-XML

Parameter [Ausblenden](#)

-- Alter der Studierenden (16340) - (gelaufen:10:31:01)

**Abschluss :**

**Aggregation Fach :** 10

**Filter Studierende :**

**Fächer :**

**Fächer-Sicht :** 2376

**Fächer-Sicht-Stand :** date\_val('17.06.2008')

**Fächer-Stand :** date\_val('17.06.2008')

**Geschlecht :**

**Hörerstatus :** 1=1

**Köpfe oder Fälle ? :** studiengang\_nr = 1 and fach\_nr = 1

**MAXOFFSET :** 30

**OFFSET :** 0

**Organigramm-Sicht :** 0

**Organigramm-Stand :** date\_val('17.06.2008')

**Seit Semester :** 20071

**Status :** 1,2,3,5,6

**Stichtag :** 0

**UserID :** 4

**bis Fachsemester :**

**bis Semester :** 20071

**erlaubt :** 0

**locale :** de

**tid :** 16340

http://mercury:8080/superx/servlet/de.superx.servlet.Entwicklung?param=maskxml

Die Parameter werden dem nun folgenden SQL-Quellcode der Abfrage übergeben (SQL-Quelle), daraus wird dann ein Script erzeugt, das nur noch die Freemarker-Befehle enthält (SQL nach 1. Transformation). Das tatsächlich nach der Freemarker-Transformation in der Datenbank ausgeführte Script wird in der rechten Spalte angezeigt.

The screenshot displays the SuperX Entwicklung web application interface. It features a navigation bar with tabs: 'Masken-XML', 'Parameter anzeigen', 'SQL-Quelle', 'SQL nach 1. Transformation', 'fertiger SQL nach Freemarker Transformation', and 'Tabellen-XML'. The main content area is divided into three columns:

- SQL-Quelle:** Shows the original SQL query for 'Alter der Studierenden (16340)'. It includes a comment: '{ diese Zwischentabelle wird erstellt, um später gew. Fälle zu ermöglichen }'. The query uses a subquery to calculate the sum of 'summe' for each student.
- SQL nach 1. Transformation:** Shows the transformed SQL code with Freemarker tags like `<@selectintemp`, `<@printkeys`, and `<@selectintemp`.
- nach Freemarker Transformation:** Shows the final SQL code after transformation, including the `group by` clause and the `create index` and `create temp table` statements.

Schlussendlich können Sie noch den XML-Code der Ergebnistabelle anzeigen (Link Tabellen-XML):

Hier die Tabelle:

[Export](#) [Druckversion](#) [XML](#) [PDF](#) [XLS](#)

#### Alter der Studierenden

Köpfe oder Fälle?: Köpfe | Stichtag: **Aktuelle Zahlen** | Seit Semester: **SS 2007** | bis Semester: **SS 2007** | Status: **Alle ohne Beur.** | Hörerstatus: **alle** | Aggregation Fach: **Fächer + Studiengänge** | User: **superx** | Stand: 28.02.2008

Datensatz 1 - 30 von insgesamt 192 Sätzen

Ebene	Studiengang	Gesamtzahl	Durchschnitt	<20	20-24	25-29	30-34	35-39	40-44	45-49	50-54	55-59	>=60	
Summe Fach (intern)		3.424,00		25,34	27,00	1.903,00	1.015,00	225,00	116,00	78,00	42,00	12,00	4,00	2,00
Fach (intern)	Biologie	139,00	24,62	1,00	86,00	39,00	8,00	3,00	2,00	0,00	0,00	0,00	0,00	0,00
Studiengang	Biologie Bachelor VM Prof.-Ord. 20052	49,00	23,22	1,00	36,00	9,00	2,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00
Studiengang	Biologie LA an Realschulen Hauptf.	1,00	26,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Studiengang	Biologie LA an Realschulen Hauptf. Prof.-Ord. 0	51,00	25,90	0,00	25,00	18,00	6,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00
Studiengang	Biologie LA an Realschulen Hauptf. Prof.-Ord. 9	10,00	25,00	0,00	6,00	3,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00

und hier der XML-Code:

The screenshot shows the XML code for the 'Alter der Studierenden' table. The XML code is displayed in a text area, showing the structure of the data table and the associated metadata. The code includes the following elements:

- `<table>` tag with attributes `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` and `xsi:schemaLocation="http://www.superx.de/tables/tables.xsd http://www.superx.de/tables/tables.xsd"`.
- `<thead>` and `<tbody>` tags containing the table structure and data rows.
- `<table>` tag with attributes `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` and `xsi:schemaLocation="http://www.superx.de/tables/tables.xsd http://www.superx.de/tables/tables.xsd"`.
- `<thead>` and `<tbody>` tags containing the table structure and data rows.

### 2.7.5.3 Berechtigung für das Entwicklungsservlet

Standard ist, dass das Servlet aus Sicherheitsgründen nur für Administratoren zugänglich ist, wenn Sie es auch als eingeschränkter User nutzen möchten, müssen Sie es für diesen User in der



tomcat/webapps/superx/WEB-INF/web.xml freigeben. Ergänzen Sie einen `<init-param>`-Block zur obig aufgeführten `<servlet>`-Definition, Inhalt ist eine mit Komma getrennte Liste der Userkennungen, die Zugriff erhalten sollen, z.B.

```
<servlet>
    <servlet-name>
        Entwicklung
    </servlet-name>
    <servlet-class>
        de.superx.servlet.Entwicklung
    </servlet-class>
    <init-param>
    <param-name>authorized_users</param-name>
    <param-value>test, test2</param-value>
</init-param>
</servlet>
```

Hinweis: Wenn Sie einen Eintrag wie oben machen, gilt der für die aufrufende URL `http://RECHNER:PORT/superx/servlet/Entwicklung`.

Nicht jedoch für die Langversion

`http://RECHNER:PORT/superx/servlet/de.superx.servlet.Entwicklung`- ggfs. also zwei `<servlet>`-Einträge für die Servletnamen `Entwicklung` und `de.superx.servlet.Entwicklung` anlegen.

## 2.7.6 Masken für das XML-Frontend vorbereiten

Das XML-Frontend arbeitet mit den vorhandenen Masken und stellt dort grundlegende Funktionen zur Verfügung. Darüber hinaus bietet das Frontend die Möglichkeit, einzelne Abfragen individuell zu gestalten. Hierzu sind allerdings grundlegende XML-Kenntnisse erforderlich.

Ein großer Vorteil des XML-Frontends ist, dass Anwender sich ihre Bericht im XML-Format herunterladen können und ohne Datenbankkenntnisse ihre Berichte "maßschneidern" können.

Es ist z.B. damit möglich, auf beliebige Berichte mit gesetzten Parametern einen Bookmark zu legen.

### 2.7.6.1 Erzeugen eines Stylesheets

Es ist möglich für Spezialfunktionen eigene Stylesheets für einzelne Masken zu hinterlegen.

Zunächst muss für das Ergebnis ein neues Stylesheet erzeugt werden. Als Vorlage für Masken können Sie das Muster-Stylesheet

```
$SUPERX_DIR/webserver/tomcat/webapps/superx/xml/maske_html.xsl
```

Für Ergebnistabellen können Sie das Muster-Stylesheet

```
$SUPERX_DIR/webserver/tomcat/webapps/superx/xml/tabelle_html.xsl
```

verwenden. Speichern Sie das Stylesheet unter einem anderen Namen im gleichen Verzeichnis ab, und ändern Sie das Stylesheet. Dann fügen Sie das Stylesheet in die Tabelle `sx_stylesheets` ein.

tid	filename	caption	description	relation	userag	contenttype
1	tabelle_html.xml	Generisches St	Generisch	table		text/html; charset=ISO-8859-1
2	tabellenfeld_bearbeiten.xml	Generisches St	Generisch	table		text/html; charset=ISO-8859-1
3	maske_html.xml	Generisches St	Generisch	mask		text/html; charset=ISO-8859-1
4	maske_html_tabfeld.xml	Generisches St	Generisch	mask		text/html; charset=ISO-8859-1
5	tabelle_html_11570.xml	Berichtsblatt K	Kurze Zusamm	table		text/html; charset=ISO-8859-1
7	tabelle_fo_rtf.xml	RTF	Export in Textv	table		application/msword
6	tabelle_fo_pdf.xml	PDF	Export in PDF (	table		application/pdf

Das Beispiel zeigt einige Styleheets, das erste ist bereits Teil des Kernmoduls, das fünfte befindet sich im COB-Modul. Zu den Feldern:

- **filename** kennzeichnet den Dateinamen relativ zum Verzeichnis  
`§SUPERX_DIR/webserver/tomcat/webapps/superx/xml`.
- **caption** dient als Kurzüberschrift, die im Ergebnisblatt als Button angezeigt wird.
- **description** stellt einen Erläuterungstext für den Button dar.
- **relation** bezieht sich auf die Beziehung des Stylesheets; mögliche Werte sind "mask" für eine Maske und "table" für Tabelle.
- **useragent** bietet die Möglichkeit, ein Stylesheet für spezielle Lesegeräte anzubieten, z.B. WAP-Handys oder Braille-Zeilen.
- **contenttype** entspricht dem useragent und kennzeichnet den `content-type`, der dem Lesegerät im http-header übermittelt werden soll. Möglich sind derzeit die obigen Varianten (svg oder excel sind in Vorbereitung).

### 2.7.6.2 Zuordnung einer Maske zu einem Stylesheet

Konkret arbeitet SuperX so: Wenn einer Abfrage ein oder mehrere Stylesheets zugeordnet sind, dann werden die in der Reihenfolge angezeigt, in der sie definiert sind. Wenn kein Stylesheet definiert ist, dann wird das Standard-Stylesheet von SuperX benutzt: `maske_html.xml` für Masken sowie `tabelle_html.xml` für Tabellen.

Die Zuordnung eines Stylesheets geschieht in der Tabelle `sx_mask_style`. Der Tupelidentifizier des Stylesheets wird in der Tabelle `sx_mask_style` im Feld `stylesheet_id` eingetragen.

Das Beispiel zeigt, dass die beiden oben beschriebenen Stylesheets der Maske 11690 zugeordnet werden.

tid	maskeninfo_id	stylesheet_id	ord
1	11690	1	1
2	11690	2	2

Das Feld `ord` kennzeichnet die Reihenfolge der anzubietenden Stylesheets. Wir sehen hier, dass zuerst das generische Standard-Stylesheet angezeigt wird, und dann das Stylesheet Nr.2.

Defaultmäßig sind die Stylesheets für html (Druckversion in neuem Fenster), xml, Excel und PDF in jeder Ergebnistabelle enthalten. Andere Stylesheets pfg müssen in der obigen Tabelle zugeordnet werden -

dies ist sinnvoll, da die Standard-Stylesheets zunächst mit der in Frage kommenden Maske erprobt werden muss.

### 2.7.6.3 Anpassung an Lesegeräte

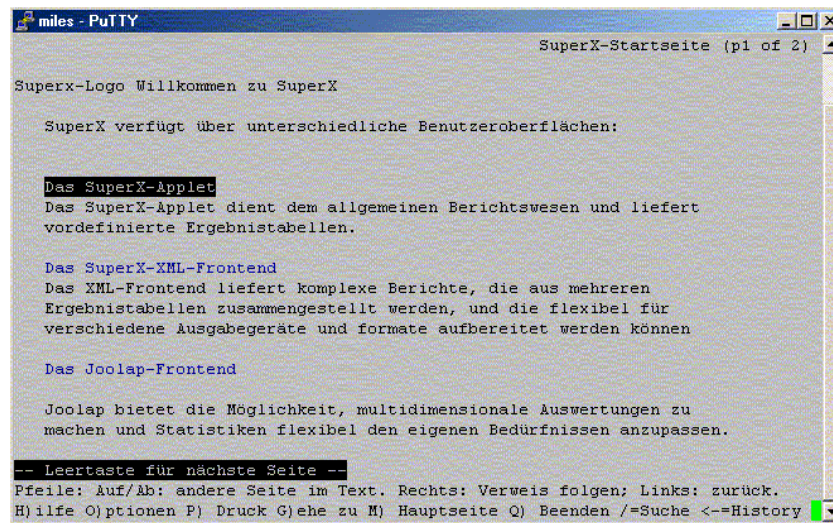
Der Vorteil von XML-Berichten ist, dass sie sich an individuelle Lesegeräte anpassen lassen. So können Sie die Standardoberfläche automatisch für das jeweilige Lesegerät anpassen und dadurch ganz individuelle Designs erzielen, z.B. auch für barrierefreie Angebote.

Das folgende Beispiel zeigt dies anhand des textbasierten HTML-Browsers **lynx**, der sich (zumindest am Anfang) gut zum Testen für barrierefreie Angebote eignet.

Klicken Sie jeweils auf die Grafik, um sie zu vergrößern.

Die rechte Abbildung zeigt die SuperX-Homepage in einer Konsole im Browser

lynx.



```

miles - PuTTY
SuperX-Startseite (p1 of 2)
Superx-Logo Willkommen zu SuperX

SuperX verfügt über unterschiedliche Benutzeroberflächen:

Das SuperX-Applet
Das SuperX-Applet dient dem allgemeinen Berichtswesen und liefert
vordefinierte Ergebnistabellen.

Das SuperX-XML-Frontend
Das XML-Frontend liefert komplexe Berichte, die aus mehreren
Ergebnistabellen zusammengestellt werden, und die flexibel für
verschiedene Ausgabegeräte und formate aufbereitet werden können

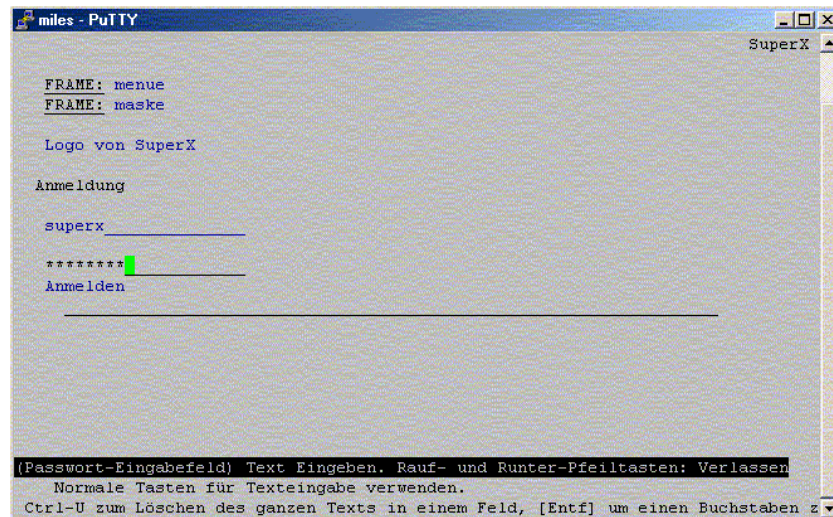
Das Joolap-Frontend

Joolap bietet die Möglichkeit, multidimensionale Auswertungen zu
machen und Statistiken flexibel den eigenen Bedürfnissen anzupassen.

-- Leertaste für nächste Seite --
Pfeile: Auf/Ab: andere Seite im Text. Rechts: Verweis folgen; Links: zurück.
H)ilfe O)ptionen P) Druck G)ehe zu M) Hauptseite Q) Beenden /=Suche <-=History

```

Wir gehen auf das XML-Frontend, und erhalten die Anmelde-  
maske. Die Frame-Tags  
ignorieren wir.



```

miles - PuTTY
SuperX
FRAME: menue
FRAME: maske

Logo von SuperX

Anmeldung

superx_____
*****█
Anmelden

(Passwort-Eingabefeld) Text Eingeben. Rauf- und Runter-Pfeiltasten: Verlassen
Normale Tasten für Texteingabe verwenden.
Ctrl-U zum Löschen des ganzen Texts in einem Feld, [Entf] um einen Buchstaben z

```

Nach erfolgreicher Anmeldung erscheint das Menü aus dem Themenbaum. Wir wählen hier als Beispiel die Abfrage **Benutzer von SuperX**.

```

miles - PuTTY
Anmeldung (p1 of 2)

SuperX-Logo

Abmelden

Willkommen John Doe

-Administration
  -Benutzer
    -Nutzungsprotokolle (intern)
    -Benutzer von SuperX
    -User einrichten
    -User löschen
  -Masken
    -Maske erzeugen
    -Maske kopieren
    -Maske löschen
    -Maske suchen
  -Felder
    -Feld erzeugen

-- Leertaste für nächste Seite --
Pfeile: Auf/Ab: andere Seite im Text. Rechts: Verweis folgen; Links: zurück.
H)ilfe O)ptionen P) Druck G)ehe zu M) Hauptseite Q) Beenden /=Suche <--History
  
```

Nun wird die Maske von dieser Abfrage angezeigt. Bei Kombinationsfeldern gehen wir auf das Feld, und drücken die Return-Taste. Es erscheinen die Auswahlinträge. Zum Abschluss gehen wir auf "Abschicken".

```

miles - PuTTY
Eingabemaske

Zur SuperX-Homepage

Benutzer von SuperX

Benutzer
Gruppe
Sachgebiete
Thema [
Administ
Archiv-Re
Abschicken

(Auswahlliste) Eingabetaste drücken und Pfeiltasten zur Optionswahl werwenden.
H)ilfe O)ptionen P) Druck G)ehe zu M) Hauptseite Q) Beenden /=Suche <--History
  
```

Es erscheint die Ergebnisanzeige. Dies sieht natürlich noch nicht besonders gut aus, weil textbasierte Browser und Tabellen sich nicht gut vertragen. Via Stylesheet lassen sich aber ganz übersichtlich Darstellungen entwerfen.

```

miles - PuTTY
Ergebnis Benutzer von SuperX (p1 of 2)

Benutzerhandbuch | Zur SuperX-Homepage

Gruppe: Administratoren ;

Benutzer von SuperX

Stand: 01.01.2003

Benutzer | Gruppe | Admini- strator | Archiv- Recht | Sachgebiete |
-----|-----|-----|-----|-----|
| Administratoren | 0 | 0 | Administration |
admin | Administratoren | 1 | 1 | Administration |
superx | Administratoren | 1 | 1 | Administration |
superx12 | Administratoren | 0 | 0 | Administration |
superx13 | Administratoren | 0 | 0 | Administration |
| | | | |
Anzahl Benutzer: | 5 | | |

-- Leertaste für nächste Seite --
Pfeile: Auf/Ab: andere Seite im Text. Rechts: Verweis folgen; Links: zurück.
H)ilfe O)ptionen P) Druck G)ehe zu M) Hauptseite Q) Beenden /=Suche <--History
  
```

Das Beispiel zeigt, dass durch XML und XSL keine Grenzen bei der Gestaltung von Benutzeroberflächen für SuperX existieren. Die obigen Stylesheets befinden sich als Muster im Verzeichnis `$SUPERX_DIR/webserver/tomcat/webapps/superx/xml`, und haben jeweils den Zusatz "html2" (für einfaches HTML Version 2.0) im Dateinamen, z.B. `maske_html2.xsl`.

Wein kleiner Tipp noch für lynx: Wenn Sie das produzierte html überprüfen wollen, dann starten Sie lynx wie folgt:

```
lynx -trace http://localhost:8080/superX/xml/
```

Eine Logdatei `lynx.trace` wird in das aktuelle Verzeichnis geschrieben.





#### 2.7.6.4 Erweiterungen des XML-Frontends

Das XML-Frontend bietet gegenüber dem Applet einige Erweiterungen, die insbesondere für aufwändiger gestaltete Webapplikationen nützlich sind:

- Die Ergebnisseiten werden nicht komplett geladen, sondern im Rahmen von frei definierbaren Intervallen, z.B. 30 Datensätze pro Seite. Am Seitenende wird dann eine Navigationsmöglichkeit geliefert (Vorherige Seite / Nächste Seite). Der Intervall wird in `$$SUPERX_DIR/webserver/tomcat/webapps/superx/WEB-INF/web.xml` definiert (Parameter `maxOffset`).
- Die Ergebnisseiten können verlinkt werden, über spezielle Navigationsspalten (s.u.).
- In Feldern können Links zu anderen Masken definiert werden (Feldart 15).

##### 2.7.6.4.1 Navigationsspalten im XML-Frontend

Wenn die Ergebnistabelle an das XML-Frontend übergeben wird, dann können spezielle Spalten für die Navigation eingesetzt werden. Die Spaltennamen werden im letzten `select` des `select_stmt` einer Makse übergeben.

<b>nexttable</b>	<p>Link auf eine andere SuperX-Tabelle; der Inhalt des Feldes wird dann um den Pfad zum Servlet, (optional auch den String der Sessionid) und den Passus "SuperXmlTabelle?tid=" ergänzt, d.h. dem Servlet wird als erster Parameter die maskeninfo-tid übergeben. So wird z.B. aus dem Inhalt: 20010&amp;id=2044 der Link http://&lt;URL der Webapplikation&gt;/servlet/SuperXmlTabelle?tid=20010&amp;id=2044</p>
<b>nextpage</b>	<p>Die Ergebnisseite wird dann um einen Button  ergänzt.</p>
<b>nextmask</b>	<p>Link auf eine andere SuperX-Maske; der Inhalt des Feldes wird dann um den Pfad zum Servlet, (optional auch den String der Sessionid) und den Passus "SuperXmlMaske?tid=" ergänzt. So wird z.B. aus dem Inhalt: 20010&amp;id=2044 der Link http://&lt;URL der Webapplikation&gt;/servlet/SuperXmlMaske?tid=20010&amp;id=2044</p>
<b>nextdelete</b>	<p>Die Ergebnisseite wird dann um einen Button  ergänzt.</p> <p>Link auf eine andere SuperX-Maske; Im Unterschied zu nextmask wird hier ein anderes Icon gewählt: Die Ergebnisseite wird dann um einen Delete-Button  ergänzt.</p>
<b>nextedit</b>	<p>Link auf ein DBForms-Formular relativ zur URL des Servlets. die Ergebnisseite wird um einen "Bearbeiten"-Button  ergänzt.</p>
<b>nextmail</b>	<p>Feldinhalte werden um einen Mailto-Tag ergänzt. z.B. info@superx-projekt.de wird zu &lt;a mailto:" info@superx-projekt.de"&gt; info@superx-projekt.de&lt;/a&gt;</p>
<b>url</b>	<p>Feldinhalte werden um einen href-Tag (sowie wenn nötig um ein "http" ergänzt. z.B. www.superx-projekt.de wird zu &lt;a href="http://www.superx-projekt.de"&gt;www@superx-projekt.de&lt;/a&gt;</p>
<b>nextlink</b>	<p>Link auf eine externe Seite oder eine andere SuperX-Tabelle; anders als bei nexttable wird ein frei wählbarer textueller Link angegeben, wobei der Volltext des Links und der eigentliche Linkt durch ein Trennzeichen " " getrennt sind. So wird z.B. der Feldwert "Erläuterungen http://www.erlaeuterungen.de" wie folgt ersetzt: &lt;a href="http://www.erlaeuterungen.de"&gt;Erl&amp;auml;uterungen&lt;/a&gt; Wenn nach dem Trennzeichen keine externe Web-Adresse angeboten wird (erkennbar am vorangestellten "http:"), dann wird der Inhalt des Feldes um den Pfad zum Tabellen-Servlet ergänzt: So wird z.B. aus dem Inhalt: Details zur Hochschule 20010&amp;id=2044 der Link &lt;a href=../servlet/SuperXmlTabelle?tid=20010&amp;id=2044&gt;Details zur Hochschule&lt;/a&gt;</p>

### 2.7.6.4.2 Hierarchieebenen in Ergebnisspalten

In Ergebnistabellen wird oft gewünscht, Tabellenüberschriften ineiner zu verschacheln. So wird z.B. aus folgender Tabelle:

Im Applet...

Themenauswahl		Maske	Tabelle									
Statistik 3-Studierende nach Alter und Geschlecht												
Parameter: Köpfe oder Fälle ? = Köpfe; Semester = WS 2004/2005; Stichtag = Aktuelle Zahlen; Fächer = Fak./FB. 05 Fak. f. Philologie (Fachbereiche und Fächer(intern)); Status = Alle; Aggregation Fach = Fächer; User=John Doe; Köpfe=Erster Studiengang, erstes Fach Stand: 29.06.2005												
Ebene	Studiengang	18-19		20-21		22-23		24-25		26-27		
		M	W	M	W	M	W	M	W	M	W	
Fachbereich	Fak./FB. 05 Fak. f. Philologie	30	309	266	1034	304	828	335	667	247	499	
Fach (intern)	Allg.&vgl. Literaturwiss.	2	8	17	59	21	51	14	25	8	16	
Fach (intern)	Allg.&vgl. Sprachwiss.	0	0	0	0	0	0	3	2	2	1	
Fach (intern)	Amerikastudien	0	0	0	0	0	6	8	21	2	2	
Fach (intern)	Anglistik	9	70	76	260	76	148	59	110	30	71	
Fach (intern)	Biling.L./L. (Zusatzstud)	0	0	0	0	0	0	0	0	0	2	
Fach (intern)	Deutsch	0	0	0	1	4	40	17	48	19	23	
Fach (intern)	Deutsch Zweitspr(Zusatz)	0	0	0	0	0	0	0	2	0	1	
Fach (intern)	Deutschuntl.Ausl.(Zusatz)	0	0	0	1	0	1	3	16	2	56	
Fach (intern)	Englisch	0	0	0	0	9	41	23	31	13	26	

Im XML-Frontend...

Export: [Druckversion](#) | [XML](#) | [PDF](#) | [RTF](#) | [XLS](#) | [Mit leeren Spalten](#)

### Statistik 3-Studierende nach Alter und Geschlecht

Köpfe oder Fälle ? : Köpfe ; Semester: WS 2004/2005 ; Stichtag: Aktuelle Zahlen ; Fächer: Fak./FB. 03 Fak. f. Philos.Päd.Publ. ; Status: Alle ; Aggregation Fach: Fächer ; Stand: 29.06.2005

Köpfe=Erster Studiengang, erstes Fach

Ebene	Studiengang	18-19		20-21		22-23		24-25		26-27		28-29		30-31		32-33		34-35		36-37		38-39		40-41	
		M	W	M	W	M	W	M	W	M	W	M	W	M	W	M	W	M	W	M	W	M	W	M	W
Fachbereich	Fak./FB. 03 Fak. f. Philos.Päd.Publ.	5	36	51	202	65	215	83	215	65	113	50	60	33	32	27	28	20	16	24	11	18	8	16	11
Fach (intern)	Erziehungswissenschaft	1	32	32	168	37	151	23	67	5	15	1	7	3	6	1	0	1	2	0	2	1	1	0	0
Fach (intern)	Europ. Culture & Economy	0	0	0	1	0	20	7	49	10	27	3	15	2	6	2	1	0	0	1	1	1	0	0	0
Fach (intern)	Philosophie	4	4	19	33	28	18	32	17	27	15	21	6	18	5	6	8	11	4	15	1	11	3	10	0
Fach (intern)	Publizistik/komm.wiss.	0	0	0	0	0	7	10	25	8	22	10	13	6	10	12	8	4	5	6	3	3	1	3	0
Fach (intern)	Pädagogik	0	0	0	0	0	19	11	57	15	34	15	19	4	5	6	11	4	5	2	4	2	3	3	0

Datensatz 1 - 6 von insgesamt 6 Sätzen.

Die Spalten werde also verknüpft. Wie geht das?

Versuchen Sie in der XIL\_PROPLIST die Spaltenüberschrift mit einem Steuerzeichen "\0002", also z.B.

**Das Steuerzeichen "\000" zur Verknüpfung von Spaltenüberschriften kommt direkt nach dem "gemeinsamen" Teil der Überschrift**

```
Column CID=1 heading_text="Studiengang" center_heading
row_selectable heading_platform readonly
width=40 text_size=60
Column CID=2 heading_text="18-19\000\n M" center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=8
Column CID=3 heading_text="18-19\000\n W" center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=8
Column CID=4 heading_text="20-21\000\n M" center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=8
Column CID=5 heading_text="20-21\000\n W" center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=8
Column CID=6 heading_text="22-23\000\n M" center_heading
row_selectable col_selectable rightJust heading_platform readonly
width=8
```

Zusätzlich kann es gewünscht sein, für diese Hierarchieebenen im Browser eine Auf- und Zuklappmöglichkeit zu haben.

Beispielsweise könnte man bei einer Abfrage „Übersicht über Kennzahlenlieferungen“ drei Spalten zu Flächeninformationen haben (2005,2006,2007) in denen angegeben wird, ob geliefert wurde:

Flächen		
2005	2006	2007
x		
	x	x

wenn man den Punkt Flächen zuklappt, soll eine Zahl erscheinen, wieviele Lieferungen es für die Jahre 2005-2007 gegeben hat:

Flächen 2005-7
1
2

Um dies zu erreichen, müssen von der Datenbank vier Spalten geliefert werden flaeche2005,flaeche2006,flaeche2007 und flaeche\_gesamt.

Der entsprechende Abschnitt in der XIL-Proplist muss so aussehen wie vorher mit Steuerzeichen \000 und allen vier Spalten



```

Column CID=2 heading_text="Flächen\000 2005" center_heading explanation=""
  row_selectable col_selectable heading_platform readonly width=15 text_size=100
Column CID=2 heading_text="Flächen\000 2006" center_heading explanation=""
  row_selectable col_selectable heading_platform readonly width=15 text_size=100
Column CID=2 heading_text="Flächen\000 2007" center_heading explanation=""
  row_selectable col_selectable heading_platform readonly width=15 text_size=100
Column CID=2 heading_text="Flächen\000 2005-7" center_heading explanation=""
  row_selectable col_selectable heading_platform readonly width=15 text_size=100

```

und jetzt kommt der Clou:

Am Ende der XIL-Proplist macht man noch eine Angabe, welche Spalten den zu versteckende Aggregationsspalten sind, also

```
hiddenAggregationColumns="Flächen\000 2005-7"
```

(Wenn es mehrere gibt, mit | getrennt angeben)

Dadurch weiß der Server, dass Flächen 2005, Flächen 2006 und Flächen 2007 Detailspalten sind und zeigt zunächst zur diese an. Wenn der Punkt Flächen zugeklappt wird, werden die Detailsspalten ausgeblendet und statt dessen wird die versteckte Aggregationsspalte Flächen 2005-7 angezeigt.

Beim Auf- und Zuklappen wird vom Server nachgeladen, dass dauert zwar einen Moment, dafür ist der Server aber informiert und auch Druckversion und Excel-/PDF-Export können angepasst werden.

Diese Funktionalität wird im XML-Frontend ausgewertet, im Applet wird das Steuerzeichen sowie `hiddenAggregationColumns` einfach ignoriert.

### 2.7.6.4.3 PDF-Export

Kurz ein paar Hinweise:

Am besten nimmt man zur Bearbeitung eine bestehende pdf-Vorlage.

Tabellen:

Für jede Spalte muss direkt unter `fo:table` ein `table-column` Knoten mit der Breite kommen (im mm)

```
<fo:table>
```

```
<fo:table-column column-width="30mm">
```

```
<fo:block font-size="10pt" text-align="start/end" font-weight="bold" font-family="serif" line-height="9pt" space-before.optimum="6pt" space-after.optimum="6pt" language="en" hyphente="true">
<xsl:value-if select="format-number(/ergebnisse..., '#.###.##0,00,'German')"/>
```

**German groß ist wichtig!!!**

```
</fo:block>
```

```
</fo:table-column>
```

```
<fo:table-header>
```

```
<fo:table-row>
```

```
<fo:table-cell border-width="0.1mm" border-style="solid" padding-left="0.5mm" padding-right="0.5mm">..
```

```
<fo:table-body>
```

```
<fo:table-row><fo:table-cell>..
```

```
<fo:block text-align="center" font-size="9pt" font-weight="bold" hyphenate="false">
```

Blöcke zusammenhalten

```
<fo:block keep-together.within-page="always">
```

Block1

Block2

```
</fo:block>
```

#### 2.7.6.4.4 Excelexport

Kurz ein paar Hinweise:

Am besten nimmt man zur Bearbeitung eine bestehende xsl-Vorlage.

Man kann eine bestehende Exceldatei als Vorlage nehmen (attribut vorlage des xls\_workbook Knotens). Dies ist praktisch, um nicht direkt erzeugbare Einstellungen zu hinterlegen, z.B.

- Skalierung auf 70 %
- wiederholende Tabellenüberschrift (Seite einrichten / Tabelle)
- Extras/Schutz/Blattschutz (Poi kann trotzdem reinschreiben!)

Wenn man

```
<xls_workbook vorlage="vorlage1.xls" removeAdditionalSheets="true">
```

Wenn man Tabellen auf Vorrat angelegt hat, kann man mit dem Tag removeAdditionalSheets=true überflüssige Tabellen entfernen.

Es werden alle Zellen neu erzeugt, man kann jedoch einzelne Zeilen oder Zellen überspringen, um in der Excelvorlage Enthaltene nicht zu überschreiben:

```
<xls_row jumpover="true">
<xls_cell jumpover="true"></xls_cell>
```

<xsl\_sheet> ist ein Tabellenblatt.

<xsl\_row> kann Attribute haben ebene=summe

Zellen

Für Zahlen <xsl\_cell style="body\_dec" numeric="true">

mögliche Attribute: width (gilt logischweise für ganze Spalte)

<sup>i</sup> Aus historischen Gründen liegen die Nummern aus Karlsruhe im Bereich 0-9990, aus Duisburg im Bereich 10000-19990.

<sup>ii</sup> Das ist grob verkürzt dargestellt, aber im Augenblick für Abfragen im Bereich Studium ausreichend. Die Prozedur ermittelt außerdem noch die Institutionen, zu denen Ein User Leserechte hat. Das versteckte Feld <<UserID>> ist in jeder Maske vor-

---

handen, und die zugehörigen Institutionen (und deren "Kinder") werden aus der Tabelle `user_institution` und `organigramm` ermittelt.

<sup>iii</sup> In anderen SuperX-Abfragen wie z.B. Studierende (allgemein) wird auch mit der Tabelle `lehr_stg_ab` gejoined, um die zugehörigen Studiengänge abzurufen - dies brauchen wir in dieser Abfrage nicht.

<sup>iv</sup> Eine Erblast des alten Win32-Client, wenn wir irgendwann wirklich mal Zeit haben widmen wir uns diesem Problem. Die Syntax ist nicht gerade elegant, ebenfalls ein Überbleibsel vom SuperX-Client 1.x (XVT-Compiler). Aus Gründen der Abwärtskompatibilität weichen wir noch nicht davon ab.

<sup>v</sup> In der Praxis würden wir nicht so arbeiten, sondern die Abfrage zunächst zu einer neuen TID kopieren, und dann ändern.

<sup>vi</sup> aus der Sprachwissenschaft – Verständigungssprache für Sprecher verschiedener Sprachen, prominentestes Beispiel: Englisch

<sup>vii</sup> Für die klassischen Organigramm-Sichten muss an Position 4 und 5 zunächst die Information `lehre(0/1)` und `erlaubt(0/1)` folgen.

<sup>viii</sup> Warum ausgerechnet Word? Das Access-Frontend ist in Visual-Basic-for-Applications programmiert, und nach unserer Erfahrung ist dies der am meisten verfügbare Editor mit VBA-Unterstützung, wenn auch Access (als Teil von MS Office) installiert ist. Der Editor WordPad z.B. bietet keine VBA-Schnittstelle. Uns war außerdem eine ausgefeilte Such- und Undo-Funktion wichtig. Theoretisch könnte man in der mitgelieferten Dokumentvorlage `editblob.dot` im gleichen Verzeichnis auch Autotexte und Makros hinterlegen. Daher: Auch wenn es ungewöhnlich ist, Word als IDE zu benutzen: nach unserer Erfahrung ist es recht praktisch. Fehlt nur noch die farbige Syntaxunterstützung...