

Database maintenance

Claudia Frie

1. ICE-Key	2
1.1 ICE-tool for modifying the ICE-key	2
1.2 Editing single characteristics and attributes	3
1.3 Editing extensive changes of characteristics and/or attributes.....	4
1.4 Deleting keys.....	5
2. Implications	5
3. Equivalences.....	7
4. Keywords	8
4.1 Updating the keywords.....	10
4.2 Maintaining the read file	11
5. Definitions.....	13

1. ICE-Key

In the database the key is organised in two tables.

The **tabellenmerkmale meta table** (table characteristics) contains all the key characteristics with their characteristic-texts.

mschluesssel	merkmal
101	study demand
102	institution stuff
103	...

The **tabellenschluesssel meta table** (table keys) contains all attributes (aschluesssel) with their no. and text (auspraegung) for the respective characteristics (mschluesssel) and other commentarial information regarding the keys, which may be used if needed. (erstes/letztes vorkommen = first/last occurrence, aenderung = change)

mschluesssel	aschluesssel	erstes vorkommen	aenderung	letztes vorkommen	auspraegung
101	1	21991	NULL	NULL	persons qualified
101	2	21991	NULL	NULL	new entrants (1. SHE)
...

Inserting and changing the key may be done directly in the database using SQL-commands or via a database front-end (e.g. phpMyAdmin). But especially for more extensive changes it makes sense to use the ICE-tool for modifying the key.

1.1 ICE-tool for modifying the ICE-key

The *Key Administration Tool (KeysAdmin)* requires a Java Runtime Environment (JRE) Version 1.5 or higher. It runs with all operating systems where Java is executed. The *KeysAdmin.jar* program may be installed on any computer by copying it into any directory of the file system. To ease the call from the prompt the best way is to copy it into a directory not too deep in the file system.

The program communicates via the socket connection with the ICE-application-server, which is launched as daemon on the server side and listens to client requests. The information regarding the database connection the application server takes from the configuration file `config/ICETab.cfg`.

To **start the application server** one changes in a console window to the directory with the ICE-server:

```
cd %TOMCAT_HOME%/webapps/iceproject/WEB-INF/classes
```

Prior to the start the path of the classes must be set:

```
export CLASSPATH=./:/path/to/jdbcdriver:/path/to/log4jjar
```

The daemon will be launched as root:

```
/path/to/java server.TabServer [portnr]
```

and by default will run on port 50.

Starting the key administration tool 'KeysAdmin'

In a console window one changes to the directory with *KeysAdmin.jar* and starts the program with:

```
/path/to/java -jar KeysAdmin.jar
```

The *ICE-Key Edit Dialog* tool for modifying keys

First the information regarding the connection with the data base is entered. Then, by using 'Connect', the existing key of the database is loaded.

Now new characteristics or new attributes for existing characteristics may be inserted, or the texts of characteristics or attributes may be changed. Single characteristics or attributes may be edited in the ICE-key overview (*ICE-Schlüsselübersicht*), or more extensive changes may be done using a read file. The *ICE-Key Overview* contains all the characteristics and attributes present in the database. Therein a characteristic is listed with its characteristic-no. and characteristic-text, the corresponding attributes with characteristic-no., attribute-no. and attribute-text.

1.2 Editing single characteristics and attributes

Inserting new characteristics

Select 'New characteristic', and enter the desired characteristic-no. and characteristic-text. With the 'Save in DB' button the characteristic will be inserted into the database.

Entry of the change into the database table:
tabellenmerkmale (table characteristics)

Adding new attributes to an existing characteristic

Choose the respective characteristic from the list, and click 'Amend characteristic'. Enter attribute-no. and text of the attribute. Insert it into the database with 'Save in DB'.

If desired, more information regarding the keys may be entered in the 'First occurrence', 'Last occurrence', and 'Changes' fields.

Entry of the change into the database table:
tabellenschluessel (table key)

Changing texts of existing characteristics and attributes

Choose the respective characteristic or the respective attribute from the list and click 'Changes'.

Depending on what was chosen from the list the text of the characteristic or attribute may be changed. (Saving in the database with 'Save in DB').

Entry of the change into the database table:
tabellenmerkmale (table characteristic) or tabellenschluessel (table key), respectively

1.3 Editing extensive changes of characteristics and/or attributes

Key changes may be entered into the database also using a read file. Rows with characteristics start with 3 tab stops, rows with attributes start with 5 tab stops. Key-no. and label texts each must be separated from each other by a tab stop. Changes of attributes always require the indication of the characteristic.

Example:
(→ = tab stop)

→	→	→	404	→	Hochschulstandorte-Namen	characteristic
→	→	→	→	→	900	→ Duisburg-Essen
→	→	→	→	→	901	→ Duisburg-Essen, U-in-E
→	→	→	→	→	902	→ Duisburg-Essen, U-in-E
→	→	→	→	→	9901	→ Lüneburg, U-in-Lünebur
→	→	→	→	→	9902	→ Lüneburg, U-Abt.-Lüne
→	→	→	→	→	9903	→ Lüneburg, U-in-Suderb
→	→	→	→	→	17200	→ U-für-Wirtschaft-und-P
→	→	→	→	→	29300	→ Dresden, H-f.-Kirchenn

Address and file name are entered into the field next to 'Read keys from file', and will be read by clicking the button.

Entry of the change(s) into the database table: *tabellenmerkmale* (table characteristic) or *tabellenschluessel* (table key) respectively

A log file with the extension *.out* informs which characteristics and attributes have been overwritten. This is always the case, when changes of existing keys are done. The log file will be filed in the same location as the read file. Inserting new characteristics and attributes as well as text changes may be done within one operation.

With the 'Load Keys' button the respective key of the database will be read again for verifying the changes done. The 'Clear fields' button resets the entries in the tool window. This command does not affect the key.

1.4 Deleting keys

Attributes and characteristics may be deleted only directly in the database (i.e. by SQL command or via a database front-end). If a characteristic shall be deleted, all corresponding attributes must first be removed from the *tabellenschluessel* (table keys) table.

With this, it must be observed that the entries in other database tables referring to the key entries to be deleted must be removed prior to deleting them from the *tabellenmerkmale* (table characteristics) and *tabellenschlüssel* (table keys) tables, respectively. Basically this applies to the following tables:

Meta tables:

Merkimp
Merkmequ
merkschlagworte
ausprschlagworte
virtbestschluessel

tabverwaltungx
sonderfehix
datenbestand
spaltenmerkmale
hinweise

Data tables:

datentab_x
edatentab_x

For the references to keys in all meta tables concerned to be updated properly, the tools described below should be used.

2. Implications

Implications represent unique hierarchical relations between keys used with the 'Sortieren' (sort) function for the table generation. Thus, for example, each higher education institution may be assigned to a type of institution (e.g. university, college of art and music, or theological university). The same applies to the functional structure: Each subject is uniquely assigned to a study area, and each study area to a subject group.

The implications are organised in the **merkimp meta table** (charimp) (merkmal = characteristic, auspr = attr).

merkmal1	auspr1	merkmal2	auspr2
125	0	123	0
125	1	123	100
125	11	123	101
125	21	123	101

For updating the implications a Java tool is available.

The `ImplGenerator` java class in the `tools` package updates the implications table in the database using a read file. The program will be started from the ICE-server home directory:

```
cd %TOMCAT_HOME%/webapps/iceproject/WEB-INF/classes
```

Prior to the start the path of the classes must be set:

```
export CLASSPATH=./path/to/jdbcdriver:/path/to/log4jjar
```

The `config/ICETab.cfg` configuration file contains information regarding the database connection and has to be adapted. The read file can be copied in an arbitrary location of the file system.

By calling

```
/path/to/java tools.ImplGenerator [path to file with implications][database account]
```

the implication update is started.

When reading, the database tables will be cleared completely and refilled with the read file. Therefore, the read file must contain all implications.

Within the **read file** the characteristics and the respective attributes are given according to the pattern

```
Merkmal1#Ausprägung#Merkmal2#Ausprägung#Merkmal3#Ausprägung  
(characteristic1#attribute#characteristic2#attribute#characteristic3#attribute)
```

wherein to or more characteristics may be set into relation to each other. In the example characteristic 1 is inferior to characteristic 2, and characteristic 2 is inferior to characteristic 3 etc.

For the following example...

	Characteristic-text	Characteristic-no.	Attribute-text	Attribute-no.
Characteristic 1	Higher education institution	405	U Hannover	14500
			U Lüneburg	09900
			FH Hannover	53300
Characteristic 2	Type of institution	504	Universities	11
			Universities of applied sciences	41
Characteristic 3	Germany	403	Land Lower Saxony	19

...these rows have to appear in the read file:

```
405#09900#504#11#403#19
405#14500#504#11#403#19
405#53300#504#41#403#19
```

Herewith it is expressed that the higher education institutions 'U Hannover' and 'U Lüneburg' are assigned to the universities, and the "FH Hannover" is assigned to the universities of applied sciences. All three higher education institution belong to the land Lower Saxony.

3. Equivalences

For equivalences relations between characteristics or characteristic attributes are established, which describe the same facts as regards content. This is particularly clear e.g. with the characteristics for higher education institutions:

- The key 'Characteristic 404' – 'Locations of institutions' contains all locations of the higher education institutions, but also attributes for the sum of the locations of the whole higher education institution.
- The key 'Characteristic 405' – 'Names of institutions' contains all higher education institutions without identifying the locations individually.

Characteristic-no.	Characteristic-text	Attribute-no.	Attribute-text
404	Locations of institutions	53300	FH Hannover (all)
		53302	FH Hannover in Hanover
		53305	FH Hannover in Nienburg
405	Names of institutions	53300	FH Hannover
		09900	U Lüneburg

The attribute 53300 of the characteristic 404 and the attribute 53300 of the characteristic 405 may be set as equivalent. (The attributes must not have the same attribute-no. or attribute-text, respectively.)

The equivalences are organised in the **merkequ meta table** (charequ) (merkmal = characteristic, auspr = attr).

merkmal1	auspr1	merkmal2	auspr2
404	13300	405	13300
404	13400	405	13400
404	13500	405	13500
...

For updating the equivalences a Java tool is available.

The `EquivalenceGenerator` java class in the `tools` package updates the *merkequ* (charequ) table in the database using a read file. The program will be started from the ICE-server home directory:

```
cd %TOMCAT_HOME%/webapps/iceproject/WEB-INF/classes
```

Prior to the start the path of the classes must be set:

```
export CLASSPATH=./:path/to/jdbcdriver:path/to/log4jjar
```

The `config/ICETab.cfg` configuration file contains information regarding the database connection and has to be adapted. The read file can be copied in an arbitrary location of the file system.

By calling

```
/path/to/java tools.EquivalenceGenerator [path to file with
equivalences][database account]
```

the equivalence update is started.

When reading, the database tables will be cleared completely and refilled with the read file. Therefore, the read file must contain all equivalences.

In the **read file** the two characteristics and then the equivalent attributes of the characteristics are given. For each information 5 characters are provided. Blanks do not have to be filled with 0 (in the example blank = _):

Merkmal1 Merkmal2 Ausprägung1 Ausprägung2
(characteristic1 characteristic2 attribute1 attribute2)

the example above would result in the following rows:

```
__405__4041450014500
__405__404_9900_9900
```

4. Keywords

With table generation or standard tables, respectively, the user searches with the keywords for the matching data stocks, using the *Search* function. By the keys the keywords are linked to the data stocks and standard tables, respectively. If links to keywords are missing. e.g. those of newly inserted keys, the stocks or standard tables might not be found with the keywords.

Example:

The keywords 'Examinations' and 'graduates' are assigned to the key 'Characteristic 101 – attribute 5' (examinations). With one of these two keywords, via the keyword search, all data stocks (or standard tables, respectively) containing data with the characteristic attribute 101-5 are offered by the database.

Here it should be observed that the data stocks of the table generation and of the standard tables are organised in two databases, the *ice* database for data tables and the *sta* database for standard tables, meaning the keywords have to be maintained in both databases.

The keywords of the *ice* database are organised in three tables.

The ***schlagworte* meta table** (keywords) contains the keywords (schlagwort = keyword, kennzahl = code number).

Schlagwort	kennzahl
Colleges	330
Professional schools	331
Subject-related semesters	332
...	...

The **merkschlagworte meta table** (charkeywords) contains the keyword assignments to the key characteristics (kennzahl = code number, mschluesel = ckey).

kennzahl	mschluesel
568	101
562	102
662	102
...	...

The **ausprschlagworte meta table** (attrikeywords) contains the keyword assignments to the key attributes (kennzahl = code number, mschluesel = ckey, aschluesel = akey).

kennzahl	mschluesel	aschluesel
792	101	1
788	101	2
788	101	3
...

The keywords of the **sta database** are organised in four tables.

The **schlagwort meta table** (keyword) contains the keywords (schlagwortnr = keywordno, schlagwort = keyword, zustand = status).

schlagwortnr	schlagwort	zustand
1	1st academic subject	1
2	2nd academic subject	0
3	3rd academic subject	0
...

In the **schluessel meta table** (keys) keys and key-no. are managed (schluesselnr = keyno, schluessel = key)

schluesselnr	schluessel
736	101#-1
988	101#1
992	101#11
415	101#12

The **schlagschluessel meta table** assigns key numbers to the keyword numbers (schlagwortnr = keywordno, schluesselnr = keyno).

schlagwortnr	schluesselnr
1	1
208	1
252	1
...	...

The **hashtabelle meta table** (hash table) contains the assignments of the keys to the standard tables (schluesselnr = keyno, tabellennr = tableno).

schluesselnr	tabellennr
1206	116
490	116
212	116
...	...

4.1 Updating the keywords

In the *ice* database:

The `KeywordGenerator` java class in the `tools` package updates the entries in the respective meta tables of the *ice* database using a read file (keywords). The program will be started from the ICE-server home directory:

```
cd %TOMCAT_HOME%/webapps/iceproject/WEB-INF/classes
```

Prior to the start the path of the classes must be set:

```
export CLASSPATH=./path/to/jdbcdriver:/path/to/log4jjar
```

The `config/ICETab.cfg` configuration file contains information regarding the database connection and has to be adapted. The read file (keywords) can be copied in an arbitrary location of the file system.

By calling

```
/path/to/java tools.KeywordGenerator [path to file with keywords][database account]
```

the keyword update is started.

When reading, the database tables will be cleared completely and refilled with the read file. Therefore, the read file must contain all keywords.

In the *sta* database:

The `StandardKeywordGenerator` java class in the `tools` package updates the entries in the respective meta tables of the *sta* database using a read file (keywords). The program will be started from the ICE-server home directory:

```
cd %TOMCAT_HOME%/webapps/iceproject/WEB-INF/classes
```

Prior to the start the path of the classes must be set:

```
export CLASSPATH=./path/to/jdbcdriver:/path/to/log4jjar
```

The `config/STATab.cfg` configuration file contains information regarding the database connection and has to be adapted. The read file keywords can be copied in the an arbitrary location of the file system. The path to the file have to be inserted in the configuration file `config/ICETab.cfg` by property `$virtualStockPath1`.

By calling

```
/path/to/java tools.StandardKeywordGenerator
```

the keyword update is started.

When reading, the database tables will be cleared completely and refilled with the read file. Therefore, the read file must contain all keywords. However, only those keywords will be read for which there are standard tables. This is checked via the *hashtabelle* (hash table) meta table. The same read file as for the *ice* database may be used.

In the read file the characteristic (5 characters width) and the attribute (5 characters width), where a keyword should be linked to, will be entered. Characteristic and attribute are separated by a blank (in the example = _). The keyword follows, separated by a comma. A keyword may be linked to multiple keys and/or a key may be linked to multiple keywords. By '-' a keyword may be linked to all attributes of the characteristic.

Example read file

```
__101_00006,Examinations
__101_00006,Graduates
__106_00035,Graduates
__142_____1,Expenditures/revenues
```

4.2 Maintaining the read file

The *schlagworte* (keywords) read file may be maintained directly 'by hand' or using a tool as needed. (Other features, such as automatically updating the keyword assignments, are not available at present.)

The keywords tool 'KeywordsAdmin' requires a Java Runtime Environment (JRE) Version 1.5 or higher. It runs with all operating systems where Java is executed. The *KeywordsAdmin.jar* program may be installed on any computer by copying it into any directory of the file system. To ease the call from the prompt the best way is to copy it into a directory not too deep in the file system.

The program communicates via the socket connection with the ICE-application-server, which is launched as daemon on the server side and listens to client requests. The information regarding the database connection the application server takes from the configuration file `config/ICETab.cfg`.

To **start the application server** one changes in a console window to the directory with the ICE-server:

```
cd %TOMCAT_HOME%/webapps/iceproject/WEB-INF/classes
```

Prior to the start the path of the classes must be set:

```
export CLASSPATH=./path/to/jdbcdriver:/path/to/log4jjar
```

The daemon will be launched as root:

```
/path/to/java server.TabServer [portnr]
```

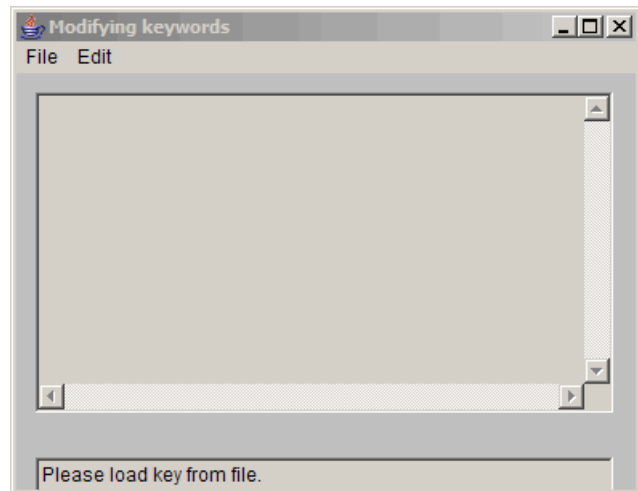
and by default will run on port 50.

Starting the keywords tool ' KeywordsAdmin '

In a console window one changes to the directory with *KeywordsAdmin.jar* and starts the program with:

```
/path/to/java -jar KeywordsAdmin.jar
```

The program requires the **GedaechtnisCode** (memory code) file. This file contains all the characteristics with the attributes and the respectively assigned keywords. Characteristic and attribute are separated by a blank, and each are 5 characters wide (in the example blank = , the attributes are filled up to 5 characters by 0s). Characteristic-text and keywords are always separated by a '#'. Particular attributes may be assigned to multiple keywords. '-1' represents all attributes of a characteristic.



The format is:

Merkmal Ausprägung#Ausprägungstext#Schlagwort1#...#Schlagwortn
(characteristic attribute#attributetext#keyword1#...#keywordn)

Example:

```
__102____-1#Schoolsystem#Schoolsystem  
__101_00006#Graduates#Graduates#Examinations
```

With these assignments the program creates the *Schlagworte* (keywords) read file. In a *warning.log* file all the rows of the *GedaechtnisCode* (memory code) are kept, in which there are discrepancies between the characteristic-texts in the *GedaechtnisCode* file and the data base. However, new characteristics or attributes (still) must be manually included into the *GedaechtnisCode*.

Creating the keyword file:

The keyword file will be created for the key of the respective sub-database using the memory code assignments.

1. Loading the key of the respective database:

'File'/'Load keys'/'from database...'

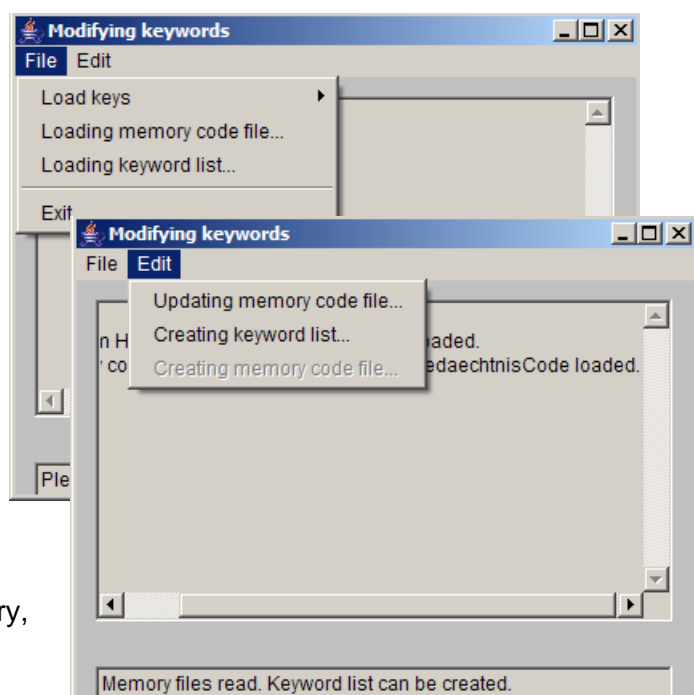
2. Loading the memory code file:

'File'/'Loading memory code file...'

3. Creating the keyword file:

'Edit'/'Creating keyword list...'

4. Saving the keyword file and, if necessary, editing the warning.log file.



Instead from the database the keyword file may be alternatively created from a key file, which has to be configured according to the file for reading key changes. (see key maintenance section)

5. Definitions

As a help for the user definitions may be inserted into the database for each keyword and for each key and may be called interactively when needed. See the corresponding manual chapter as well. (For the newly developed table generation this functionality is not yet installed.)

A definition consists of the term, the definition source, the definition text and, if needed, of synonyms. They are managed via four corresponding database tables:

The ***begriffe meta table*** (terms) contains a consecutive number of the terms and the terms themselves (begriffid = termid, begriff = term).

begriffid	begriff
1	Type of study
2	(Post)graduate studies
3	Studies
4	...

The ***defquellen meta table*** (defsources) contains a 'list' of the definition sources used (quelleid = sourceid, quelle = source).

quelleid	quelle
1	StBA (Federal Statistical Office)
2	Office of the conference of cultural ministers
3	HIS

The ***definitionen meta table*** (definitions) contains the definition texts and assigns to those a term (via *begriffid*, termid) and a source (via *quelleid*, sourceid).

begriffid	quelleid	definition
1	3	e.g. basic studies, first degree course,...
2	3	In the ICE-key system the aggregation...
3	1	University education and studies impa...
4	1	...

The **synonyme meta table** (synonyms) assigns synonyms to each term via the *begriffid* (= termid, sinonim = synonym).

begriffid	sinonim
4	Type of qualification
4	Qualification
7	Students
7	Students (registered students)
14	Type of institution
14	Type of institution (diff.)
...	...

The specification of synonyms is important, since the definition call is tied to the exact character sequence of the terms, and many similar keys or keywords require the same definitions. An example for clarification: To search for stocks with university staff by keywords, the keywords 'university staff' or 'staff, university' are possible. Both the keywords should deliver the same definition, but only one term may be assigned to a definition text. In order that the definition still can be found, any number of synonyms may be assigned to each term (via the *begriffid*, termid). Vice versa it is possible to assign multiple definition texts (e.g. with different sources) to one term by synonyms.

For maintaining the definitions no tools are available so far. Therefore, inserting and maintaining the definitions has to be done using a database front-end (e.g. phpMyAdmin) or by SQL-commands. When inserting new definitions it makes sense to adhere to a certain order of the substeps.

Steps for reading definitions:

1. The *begriffid* (termid) will be given as a consecutive number. Therefore, for reading a new definition a free number has to be detected (begriffe = terms).

```
SQL> select begriffid from begriffe order by 1;
```

2. The term is entered with the determined number

```
SQL> insert into begriffe values(211,'testterm');
```

3. and, if necessary, a new source accordingly (quelleid = sourceid).

```
SQL> insert into quelleid values(3,'HIS');
```

4. Inserting the definition text (definitionen = definitions).

```
SQL> insert into definitionen values(211,1,'definitiontext')
```

Line breaks may be created by Inserting '/n/r' at the corresponding position.

5. Inserting the synonyms (for each synonym individually) (synonyme = synonyms)

```
SQL> insert into synonyme values(211,'testsynonym');
```

Steps for deleting definitions (reverse order compared to above)

1. Deleting a synonym

If all synonyms shall be deleted (synonyme = synonyms, BegriffID = TermID):

```
SQL> delete from synonyme where BegriffID=...
```

if only particular synonyms shall be deleted (sinonim = synonym):

```
SQL> delete from synonyme where BegriffID=... and sinonim='...';
```

2. Deleting the definition text (definitionen = definitions, quelleid = sourceid)

```
SQL> delete from definitionen where begriffID=... and quelleid=...;
```

3. Deleting the term (begriffe = terms)

```
SQL> delete from begriffe where begriffID=...;
```

Updating a definition text (definitionen = definitions, Absatz = paragraph, begriffid = termid, quelleid = sourceid)

```
SQL> update definitionen set definition='Absatz1\n\rAbsatz2' where begriffid=211 and quelleid=1;
```