Dbengine

Ein System zum Verwalten und Benutzen von Datenbanken
über die Web-Schnittstelle
Version 2.3.0

von
Detlef Dürr

3. Ausgabe März 2005

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

1.1 Vorbemerkung

Das PostgreSQL-Hilfstool DBENGINE ist ein Werkzeug zur Bedienung und Administration von Datenbanken unter PostgreSQL über die Web-Schnittstelle. Mit Hilfe dieses Tools können existierende Datenbanken in einem Intranet oder auch über das Internet genutzt werden. Eines ist dabei allerdings zu beachten: Z. Z. ist das Tool nur ohne Verschlüsselung, also nur mit http und nicht mit https, getestet.
Dieses Handbuch wendet sich vorrangig an Administratoren, da weniger die Bedienung einer Datenbank als die Konfiguration dieser Bedienschnittstelle beschrieben wird. In einem ersten Teil wird dennoch kurz auf das Aussehen der Benutzerschnittstelle eingegangen, da eine sinnvolle Konfiguration nur möglich ist, wenn man weiß, unter welchen Anwendergesichtspunkten die Schnittstelle grundsätzlich vordefiniert ist. Die Kenntnis von PostgreSQL und der Sprache SQL wird vorausgesetzt und Textstellen, die sich auf dieses Datenbanksystem beziehen werden nicht weiter erläutert.
Im zweiten Teil erfolgt dann eine vollständige Beschreibung der Hilfsmittel zur Konfiguration von Datenbanksystemen, die dem Anwender zur Verfügung gestellt werden sollen. Die gesamte Konfiguration geschieht über die PostgreSQL-Datenbank 'dbengine_info_db', die selbst auch mit dem Tool bearbeitet werden kann.
Das Tool geht von einer Client-Server-Architektur aus und ist vollständig in Perl geschrieben. Die eigentliche Bearbeitung findet auf dem Server statt, während der Client als ein 'Thinclient' nur zur Anzeige und Entgegennahmen der Daten dient.
Zur Benutzung des Tools ist daher serverseitig ein Rechner mit einem Perlinterpreter nötig. Zur Anzeige der Daten und zu ihrem Handling wird die Browserschnittstelle des Internet benutzt. Daher ist auf der Serverseite das Vorhandensein eines Webservers (z.B. Apache) nötig. Die Perlskripte laufen als CGI-Skript des Webservers.
Als Schnittstelle zu Datenbanken wird die DBI-Schnittstelle von Perl benutzt. Daher kann das Datenbanksystem auf einem anderen Rechner laufen und muss nicht auf dem Webserver installiert sein. Vom Datenbanksystem wird also nur die Datenbankclientschnittstelle benötigt. Auch können grundsätzlich andere Datenbanksysteme genutzt werden. Allerdings ist dann der Perl-Code an einigen Stellen zu ändern, da auch spezielle Funktionen vom Datenbankmoduln DBD::PG benutzt werden, und da an einigen Stellen Besonderheiten von PostgreSQL genutzt werden. Auf einem PostgreSQL-Server muss außerdem eine Konfigurationsdatenbank 'dbengine_info_db' existieren. Zwar läuft das System auch ohne jegliche zusätzliche Konfiguration, aber dann nur in einer nicht optimalen und eingeschränkten Form, da einige Funktionen (z.B. Berichte) nur nach entsprechender Konfiguration lauffähig sind. Auch Querverweise von einer Tabelle zu einer anderen sind nur nach einer entsprechenden Konfiguration möglich.
Auf der Clientseite wird lediglich ein Webbrowser benötigt, da auf ihr – wie oben schon gesagt – nur die Anzeige und Entgegennahme von Eingabedaten erfolgt. Da normalerweise auf allen heutigen Systemen ein Webbrowser vorhanden ist, ist auf der Clientseite keine weitere Installation nötig.
Das Tool gibt seine Texte sprachabhängig aus. Welche Sprache es benutzt, wird anhand der vorhandenen Sprachen (z.Z. Deutsch und Englisch) und der Browsereinstellung entschieden. Bei den hier gezeigten Abbildungen war stets die Sprache Deutsch (de) als Vorzugssprache im Browser eingestellt.
Da die Webschnittstelle zustandsfrei arbeitet, empfiehlt es sich, bei jeder Tabelle einer Datenbank, die von mehreren Benutzern geändert werden kann, ein zusätzliches Feld mit dem Feldnamen 'tmin' einzuführen und diesem Feld den Datentyp 'timestamp' zu geben. Über dieses Feld, sofern

vorhanden, wird überprüft, ob der Datensatz nach dem Einlesen durch einen Benutzer von einem anderen geändert wurde und deswegen von dem überprüften Benutzer nicht geändert oder gelöscht werden darf.
Mit Hilfe des Tools kann jede PostgreSQL – Datenbank den Benutzern über das Web zur Verfügung gestellt werden. Dazu bedarf es keiner Datenbank spezifischen Konfiguration. Diese aber ist möglich, um damit das Aussehen auf dem Bildschirm den Erfordernissen besser anpassen zu können. In der Konfiguration kann auch festgelegt werden, wer welche Datenbanken bearbeiten darf.

1.2 Installation

1.2.1 Installationvorgehen

Geliefert wird ein gepacktes 'tar'-File mit allen notwendigen Dateien zur Installation von DBENGINE. Als erstes muss diese Datei mit dem Kommando gunzip in einem zusätzlichen Verzeichnis entpackt werden. Dabei entstehen zwei Unterverzeichnisse:

- Das eine, dbengine-2.3.0, enthält alle Dateien der DBENGINE-Maschine.
- Das andere, contrib, enhält eine korrigierte Version des DBD::Pg – Moduls 1.21 und die Version 0.39 von Math-Currency.

Nach dem Entpacken kann die Installation erfolgen. Zur Installation ist es nötig als 'root' im System zu arbeiten. Die Installation kann auf zwei Weisen geschehen:

- Durch Aufruf des Kommandos 'install.sh' im Unterverzeichniss 'dbengine-2.3.0' oder
- durch manuelles Ausführen von Kommandos zur Installation.

Die Installationsroutine erfragt fünf wesentliche Merkmale – sofern sie nicht als Parameter mitgegeben wurden –, um dann die Dateien an die richtigen Stellen zu kopieren:

1. Parameter '-c': das Verzeichnis für das CGI-Script (default: /usr/local/httpd/cgi-bin)
2. Parameter '-d': das DBENGINE-Dateien Verzeicnis (default: /usr/local/dbengine)
3. Parameter '-n': der Name des postgreSQL Datenbankservers (default: localhost)
4. Parameter '-p': der Port des postgreSQL Datenbankservers (default: 5432)
5. Parameter '-a': der Name des postgreSQL-Administraors, i.e. Hauptbenutzer für die dbengine_info_db Datenbank (default: postgres).

Diese Eingaben werden von der Installationsroutine verarbeitet und daraus auch das eigentliche Startscript 'dbengine.cgi' erstellt. Bei einer manuellen Installation ist auch dieses Script manuell zu erstellen. Es empfiehlt sich daher, die Installationsroutine zu benutzen.
Nach dem Ablauf von 'install.sh' sind in der Regel noch einige Einträge in der 'dbengine_info_db'-Datenbank zu korrigieren. Zur Benutzung der Datenbanken müssen die vorhandenen 'default'-Werte geändert werden. Da dann das neue Tool noch nicht einsatzfähig ist, müssen diese Änderungen über das Kommando 'psql' durchgeführt werden. Gleiches gilt für die Datenbanken aus dem Verzeichnis 'test': 'dbengine_test_db' und 'dbengine_test2_db, sofern sie benutzt werden sollen. Zusätzlich ist im Verzeichnis 'test' noch das Schema einer 'ereignisse' Datenbank, für die spezielle Aktionen programmiert wurden, vorhanden.
Als erstes muss die Tabelle 'configs' dem aktuellen System angepasst werden, um die Liste der Benutzernamen, die als 'admin' arbeiten dürfen zu setzen bzw. zu korrigieren. Um die Liste der 'admin'-User zu erhalten, kann das Kommando

```
SELECT * FROM configs WHERE index = 'admins';
```

genutzt werden. Normalerweise trägt das Installationsskript den vorgegebenen Administrator in die Datenbank ein. Ist es eine leere Liste oder fehlt der richtige Administrator, so sind die aktuellen 'admins' mit 'INSERT" sonst mit folgendem 'UPDATE'-Kommando einzutragen:

```
UPDATE configs SET values='<list of comma-separated-names>'
WHERE index='admins';
```

Ebenso muss die Tabelle 'dbengine_user' geändert werden, um so die für die jeweiligen Datenbanken zulässigen Benutzer anzugeben. Wenn für eine Datenbank kein Benutzer eingetragen ist, so kann grundsätzlich jeder Benutzer (aus Sicht des Tools) diese Datenbank benutzen. Die Nutzungseinschränkungen liegen dann aber immer noch auf der Ebene von PostgreSQL. In jedem Fall aber werden die Datenbanken dem Benutzer angezeigt. Die Änderung geschieht analog der für die 'admins'. Im folgenden ein kleines Beispiel dafür:

```
SELECT * FROM dbengine_user WHERE dbasename='<name of the
database>';
```

So werden die vorhandenen User einer Datenbank angezeigt. Diese können dann mittels eines 'UPDATE' geändert werden, oder, falls kein Eintrag vorhanden ist, kann mit 'INSERT' ein entsprechender angelegt werden. Das 'UPDATE'-Kommando würde dann lauten:

```
UPDATE dbengine_user SET username WHERE dbasename='<name of
the database>';
```

Normalerweise ist wenigstens ein Satz schon in der Tabelle vorhanden, der nur für die 'admins' den Zugriff auf die Datenbank 'dbengine_info_db' gestattet.
Zusätzlich zu

1.2.2 Resourcen

1.2.2.1 Benötigte Perl-Resourcen

Zum Lauf des dbengine-Tools wird folgendes PERL-System benötigt:

1. PERL > Version 5.00
2. PERL-MODULES:
    - Mcrypt (zur eigenen Installation dieses Moduls, wird das Paket libmcrypt incl. Der Quellen benötigt)
    - MIME::Base64
    - Math::Currency
    - DBI (nur für DBD::Pg getestet)
    - DBD::Pg (zumindest die Version 1.21, besser 2.02 oder größer)
    - CGI

Die DBD::Pg Versionen 1.21 und 1.22 haben einige Fehler (s. Datei 'README'). Einige davon sind in der im tar-File mitgelieferten Version (1.21) korrigiert, ein weiterer Fehler wird durch eine zusätzliche Konfigurationsoption umgangen.

1.2.2.2 Sonstige Resourcen

Zur Nutzung des DBENGINE-Tools müssen noch folgende zusätzliche Bedingungen erfüllt sein:

- Es muss ein System mit einem Webserver vorhanden sein, auf dem das Tool laufen kann.
- Es muss ein PostgreSQL-Server von diesem System aus ansprechbar sein.
- Dem PosgreSQL-Server müssen alle Nutzer des Tools mit ihrem Benutzernamen und dem Passwort bekannt und in der 'pg_hba.conf' muss zumindest folgende Erlaubnis – sofern der PostgreSQL-Server auf dem gleichen Rechner läuft, wie der Web-Server – eingetragen sein:

```
local   all     all                                     trust
host    all     all     127.0.0.1       255.255.255.255 trust
```

1.2.2.3 Test des Systems

Bei Erweiterungen oder auch spezifischen Einträgen in der Konfigurationsdatenbank in Form von Perl-Prozeduren ist es meist nötig, diese zu testen. Der Test in seiner Ausführlichkeit wird über einen Debuglevel in der Variablen '$debug' im Kopf des Moduls 'dbengine.pl gesteuert:

Level	Wirkung
0	Keine Testausschriebe
1	Testausschriebe erfolgen in jedem Fall im Text der erzeugten HTML-Seite. Ein Beispiel dafuür sind die SQL-Kommando-Ausschriebe beim Erzeugen eines Listings am Anfang des Listings.
2-10	Erhöhung der Insensität der Ausschriebe, je höher der Level, um so mehr Ausschriebe werden erzeugt. Durch Korrektur der Variablen '$debug_co' im Kopf des Moduls 'dbengine.pl' kann gesteuert werden, ob die Ausschriebe direkt erfolgen oder als Kommentar.
11	Höchster Level, dient auch zum Test der Apachekopplung ..., Alle Ausschriebe werden in eine Testdatei '/tmp/..db_debug.$$' geschrieben.

Dazu stehen Prozeduren zur Erzeugung von Testausschrieben zur Verfügung (s Tabelle 1.1):

Name	Aufrufparameter	Bedeutung
dbprint	(debuglevel, format, par1, par2, ...)	Format ist ein Perlformat, das mit Parametern par1, ... versorgt werden kann.
dbprint_cmd_binds	(debuglevel, cmd, bv1, bv2, ...)	Ausgabe des SQL-commands cmd mit den zugehörigen Bind-values bv1, ...
dbprint_hash	(debuglevel, name, rhash)	Ausgabe eines Hash-Arrays mit der Referenz 'rhash' unter dem Namen name.
dbprint_hash_hash	(debuglevel, name, rhash)	Ausgabe von Hash-Arrays, die unter den Schlüsseln des Hash-Arrays gefunden werden.
dbprint_array_hash	(debuglevel, name, rarray)	Ausgabe von Hash-Arrays, die unter der Array-Referenz 'name' gefunden werden.

Tabelle 1.1Tabelle der Debug-Prozeduren

2 Der Aufbau des Bildschirms

In diesem Kapitel wird das grundsätzliche Aussehen der Clientseite beschrieben. Alle Daten, die vom Server kommen, werden auf einem Fenster des Webbrowsers dargestellt. Hier werden die verschiedenen Fenster beschrieben.

2.1 Das Startfenster

Um das Tool zu starten, muss auf der Clientseite im Webbrowser in der Regel folgende URL angegeben werden:

```
http://<Servername>/cgi-bin/dbengine.cgi
```

Das Perlmodul 'dbengine.cgi' auf dem Server dient für alle Aufrufe als Verbindung zum Datenbanksystem. Der Server antwortet dann mit dem Startfenster. Sein Aussehen ist in Abbildung 2.1 dargestellt.

Abbildung 2.1 Startbild

Vom Benutzer werden die Eingabe seines Zugangsnamens und seines Passwortes, unter dem der Benutzer Zugang zur gewünschten Datanbankl hat, erwartet. Außerdem kann er aus einer Liste (dem Popupmenu) die Datenbank, die er bearbeiten möchte, auswählen. Welche Datenbank(en) er bearbeiten darf, kann der Administrator in der Tabelle 'Configs' der Konfigurationsdatenbank einstellen. Mit dem Drücken des OK-Buttons startet der Anwender das Datenbanksystem. Da das System nahezu zustandsfrei arbeitet, ist ein Ausloggen nicht nötig. Nach einiger Zeit der Nichtbenutzung (einstellbar ebenfalls über die Tabelle 'Configs') werden die noch vorhandenen Anwenderdaten aus dieser Sitzung auf dem Server automatisch gelöscht.
Ist eine Datenbank mit mehreren Schemata angewählt, so muss durch den Administrator in der Datenbankspezifischen Konfigurationsdatei die Menge der Schemata angegeben werden, die der Benutzer bearbeiten darf. Ist keine Eingabe erfolgt, so gilt – sofern der Benutzer die Datenbank aufrufen darf – der Schemastandardwert: 'public, $user', wie er in dem Datenbanksystem postgreSQL als 'default'-Wert eingetragen ist. Andernfalls erscheint ein Fenster zur Auswahl der Schemata, die der Benutzer in dieser Sitzung anwenden möchte:

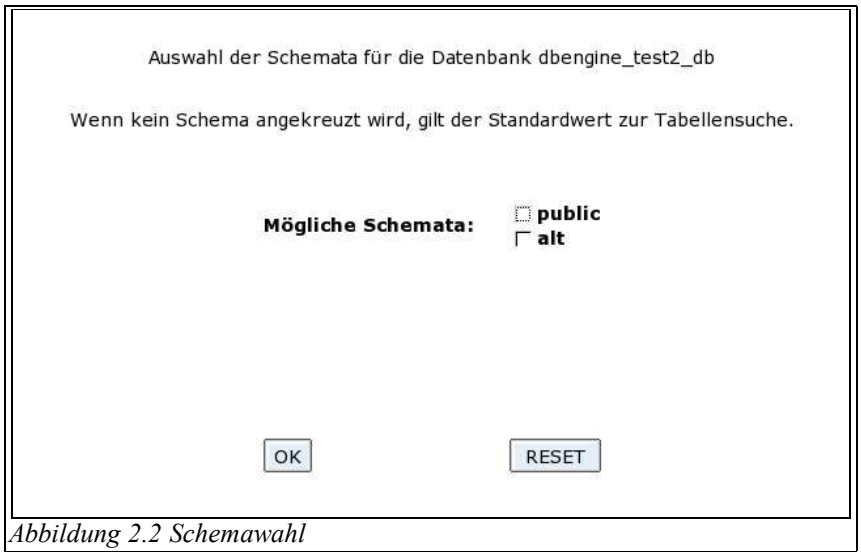


Abbildung 2.2 Schemawahl

Dabei kann auch mehr als ein Schema ausgewählt werden. Wird keines gewählt, so gilt die postgreSQL-Voreinstellung.

2.2 Das Arbeitsfenster

Nach der Eingabe der Benutzerdaten antwortet das System mit dem Arbeitsfenster. Dieses ist in zwei Bereiche aufgeteilt: die Steuerleiste und den Arbeitsbereich. Die Steuerleiste dient der Auswahl der zu bearbeitenden Tabelle der gewählten Datenbank und auf dem Arbeitsbereich werden die Daten der ausgewählten Funktion dargestellt.

2.2.1 Die Steuerleiste

Die Steuerleiste nimmt als eigener Frame die linke Seite des Arbeitsfensters ein und ist selbst wiederum in mehrere Bereiche unterteilt. Zunächst werden alle zugreifbaren Tabellen – unter Umständen in mehreren Gruppen – aufgelistet. Für Tabellen, die nicht zum Schema 'public' gehören, wird als Tabellenname die übliche Kopplung von Schemaname und Tabellenname genutzt. Sind Tabellen über die Basiskonfiguration in der Konfigurationsdatenbank definiert, so werden sie in vorgegebenen Gruppen einsortiert und, sofern gewünscht, mit dem vorgegebenen – für den Benutzer besser assoziierbaren Titel versehen – eingetragen.
In der Abbildung 2.3 sind die Aufteilungen der Steuerleiste für die Beispieldatenbanken 'dbengine_test_db' und 'dbengine_test2_db' abgebildet. Man sieht, dass die Tabellen in beiden Fällen in mehrere Bereiche unterteilt sind:

dbengine_test_db	dbengine_test2_db
• Grundformulare • Details • Wertebereiche	• Grundformulare • Tabellen

Diese Unterteilung der Tabellen ist, wie gesagt, eine Angelegenheit der Konfiguration. In der Tabelle 'Basedesign' der Konfigurationsdatenbank werden diese Daten eingetragen. Alle Tabellen einer Datenbank, die nicht in der Konfiguration eingetragen sind, werden schließlich unter einer zusätzlichen Rubrik, Tabellen, mit ihrem Tabellennamen, gegebenenfalls inklusive dem Schemanamen, eingetragen. Die Unterscheidung der Gruppen dient nur der Übersichtlichkeit, intern werden alle Tabellen gleich behandelt.

Abbildung 2.3 Die Steuerleiste

Im unteren Teil der Steuerleiste sind noch zwei weitere Bereiche vorhanden. Es sind

- Aktionen und
- Berichte.

Zusätzlich zu den Tabellen werden in diesen Gruppen die dem Benutzer zur Verfügung stehenden Datenbankaktionen und Berichte aufgeführt. Sie können über die Konfigurationsdatenbank für die Benutzer unterschiedlich freigegeben sein. Die Trennung in Aktionen und Berichte ist wie die Aufteilung der Tabellen eine Aufgabe der Konfiguration. Die Methode der Bearbeitung ist in beiden Fällen gleich: es werden durch die Konfiguration festgelegte Funktionen gestartet. Die Konfiguration geschieht über die Tabelle 'Actions' der Konfigurationsdatenbank.

2.2.2 Die Eingabemaske

Wählt der Anwender aus der Steuerleiste eine Tabelle aus, so erscheint die Eingabemaske. Sie dient drei Aufgaben:

- Der Suche nach Sätzen in der gewählten Tabelle,
- Der Neueintragung von Sätzen in der Tabelle und
- Der Änderung einzelner Sätze der Tabelle.

Die Eingabemaske ist dazu in vier Bereiche unterteilt, die Eingabefelder, die Funktionsknöpfe, einen Querverweisbereich auf andere Tabellen und ein Eingabefeld für einen eigenen speziellen SELECT-Befehl. Diese Bereiche werden nun im Einzelnen kurz erläutert. Weitergehende Erläuterungen über

die Bedeutung der Bereiche ergibt sich auch aus den Masken der Beispieldatenbank 'dbengine_test_db'.

2.2.2.1 Die Eingabefelder

Die Eingabefelder dienen verschiedenen Aufgaben: der Suche nach Sätzen aus der Datenbank (Suchmaske) oder dem Eintrag neuer bzw. der Änderung vorhandener Datensätze (Ergebnismaske). Für jedes Tabellenfeld der ausgewählten Tabelle ist in der Regel ein Eingabefeld vorhanden. Die Anordnung und die Art der Bearbeitung der Felder geschieht über die Konfigurationstabelle 'desinginfo'. Sind Felder dort nicht definiert, so werden sie an das Ende der Eingabefelder gestellt und dort entsprechend ihres Datentyps für Eingaben vorgesehen.
Neben den Tabellenfeldern können in diesem Bereich auch berechnete, virtuelle, Felder erzeugt werden. Ihr Aussehen und ihre Berechnung erfolgt über die Konfigurationsdatei 'virtual' zusammen mit der Beschreibung in 'designinfo'. In letzterer Datei haben diese Felder stets den Datentyp 'VIRTUAL'.
Sollen einzelne Felder nicht ausgegeben werden, so bekommen sie in 'designinfo' den Datentyp 'NONE'.
Die anfängliche Eingabemaske enthält normalerweise noch keine Daten (eine Ausnahme können durch die Konfiguration festgelegte 'Default'-Werte sein). Es erscheint also ein Bild der Eingabefelder mit leeren Eingabemasken. Auch die virtuellen Felder enthalten noch keine Daten, da Werte für diese Felder ja normalerweise in Abhängigkeit von den Werten eines Datensatzes errechnet werden. In der Abbildung 2.4 ist eine Suchmaske für die Beispieldatenbank dargestellt. Im Popupfeld für das Feld 'Supplier' ist schon ein Wert aus der vorhandenen Liste möglicher Lieferanten aus der Tabelle Supplier ausgewählt. Für das Popupfeld wurde aus der Liste die Zahl 6 und in Klammern den Wert 'Intel' ausgewählt. Dies geschieht durch die Verklammerung von Wert und Beschreibung in der Konfiguration über die Konfigurationstabelle 'designinfo'.

Abbildung 2.4 Beispiel einer Suchmaske

Wird in dieser Maske nun der Button 'Suchen' betätigt, so wird in der Datenbank nach Sätzen gesucht, die das Suchmuster befriedigen. Dabei können in einfachen Eingabefeldern auch neben den Werten auch Bedingungsoperatoren angeben werden. Welche Bedingungsoperatoren angegeben werden können, ist Datentyp spezifisch. Die möglichen Operatoren und die zugehörigen Datentypen sind in der Tabelle 2.1 aufgelistet:

Operator	Symbol	Zulässige Datentypen
Gleich	'="	Char, Text, Int, Float, Numeric, Date
Ungleich	'!=', '<>'	Char, Text, Int, Float, Numeric, Date
Kleiner	'<'	Int, Float, Numeric, Date
Kleiner oder gleich	'<='	Int, Float, Numeric, Date
Größer	'>'	Int, Float, Numeric, Date
Größer oder gleich	'>='	Int, Float, Numeric, Date
Match case-sensitiv (regular expression)	'~'	Char, Text
Match case-insensitiv	'~*'	Char, Text
Not match case-sensitiv	'!~'	Char, Text
Not match case-insensitiv	'!~*'	Char, Text
Entsprechend dem LIKE-Operator	'~~'	Char, Text
Entsprechend NOT LIKE	'!~~'	Char, Text

Tabelle 2.1 Tabelle der Bedingungsoperatoren

Die Bedingungsoperatoren können nur in Feldern eingegeben werden, die eine normale Eingabe zulassen. Nach der Eingabe der Bedingungsoperatoren können beliebig viele Leerstellen vor der eigentlichen Bedingung folgen. Die Bedingung selbst kann, wie bei SQL in vielen Fällen nötig, in „'“ eingeschlossen sein, ist aber nicht erforderlich. Nach einem Operator werden die einschließenden „'“ entfernt. Für einige Datentypen gibt es noch Besonderheiten zu beachten:

- Datumsfelder sind 10 Zeichen breit. Damit trotzdem Bedingungsoperatoren eingegeben werden können, können zum einen 25 Stellen eingegeben werden – nötig für eine Bereichseingabe – und außerdem kann die Jahreszahl zweistellig angegeben werden. Dann bedeuten Werte über 45 Jahre ab 1900 und Werte unter 46 sind Jahre ab 2000. Außerdem können in Datumsfeldern spezielle Datenbereiche angegeben werden. Diese Möglichkeiten zeigt die Tabelle Tabelle 2.2.

Angabe	Bereichswerte	Beispiel Eingabe	Beispiel Wertebereich
mm.jj oder mm.jjjj (1<=mm<=12)	Monat mm des Jahres jj	06.02	>= 01.06.2002 AND < 01.07.2002
Qx.jj oder Qx.jjj (1<=x<=4)	Quartal x des Jahres jj	Q2.98	>= 01.04.1998 AND < 01.07.1998
Hy.jj oder Hy.jjj (1<=y<=2)	Halbjahr y des Jahres jj	H2.99	>= 01.07.1999 AND < 01.01.2000
Jj oder jjj	Jahr jj	96	>= 01.01.1996 AND < 01.01.1997
[d1 – d2]	Zeitraum von Datum d1 bis Datum d2	[01.02.78-31.10.79]	>= 01.02.1978 AND <= 31.10.1979

Tabelle 2.2 Tabelle möglicher Datumbereichsangaben

- In Character- oder Textfeldern eingegebene Werte werden standardmäßig als 'Case-insensitiv-regular-expression' aufgefaßt. Ist allerdings in dem Feld ein '%' oder ein '_' eingegeben, so wird die Eingabe eines SQL-Jokerzeichens vermutet und der Vergleich mit dem Operator 'LIKE' durchgeführt. Soll eine Identitätsprüfung erfolgen, so muss als erstes Zeichen der Identitätsoperater '=' mit eingegeben werden.
- Wird eine Auswahl aus einem 'Multiplepopupfenster' als Suchbedingung gewählt, so werden alle ausgewählten Einträge als eine Oder-Bedingung aufgefasst und daher im Select-Statement mit 'OR' verknüpft. Eine Angabe von Bedingungen in mehreren Feldern hingegen wird als Und-Bedingung aufgefasst, und dementsprechend werden die einzelnen Suchkriterien im Select-Statement mit 'AND' verbunden.

Sollen Bedingungen zu Suche angegeben werden, die durch Eingabe in den einzelnen Feldern nicht erfüllt werden können, so kann im Feld 'Extra Suchbefehl' ein SQL-SELECT-Staement angegeben werden. Dies unterliegt einigen Einschränkungen: im 'FROM'-Teil der Anweisung muss als erste Tabelle die Tabelle auftreten, deren Maske angewählt wurde. Auch darf kein 'Subselect' in der Anweisung benutzt werden.
Gleichgültig, wie die Suche erfolgen soll, ob über die Eingabe in den Eingabefeldern oder über einen gezielten SQL-SELECT: Ein Betätigen des 'Suchen' – Buttons bewirkt dann eine Suchen nach Datensätzen, die dann entweder als Liste von Tabellensätzen (Abbildung 2.6 in Kapitel 2.2.3) oder als Ergebnismaske (Abbildung 2.5) erscheinen. Werden mehrere Sätze gefunden, so wird eine Liste von Tabellensätzen erzeugt. Wird in der Datenbank nur 1 Satz gefunden, der die Suchbedingen erfüllt, so erscheint er in der Ergebnismaske. Dies kann für eine Datenbank dadurch verhindert werden, dass für diese Datenbank in der Konfiguration der Wert 'single_sentence' auf 'true' gesetzt wird.
Will man keine Sätze suchen, sondern nur neue anlegen, so kann dies durch Eintragen der Werte in die Tabellenfelder geschehen und durch anschließendes Betätigen des 'Anlegen' – Buttons.
Die Ergebnismaske hat im ersten Teil den gleichen Aufbau wie die Suchmaske. Nur sind jetzt die Eingabefelder ausgefüllt, soweit Daten in der Datenbank für diesen Satz vorhanden sind. Auch erlaubt die Ergebnismaske mehr Operationen. Aber die Felder können wie bei der Suchmaske beliebig verändert werden und für eine erneute Suche genutzt werden.
In der Ergebnismaske sind häufig auch berechnete Felder vorhanden, Felder vom Konfigurationstyp 'VIRTUEL', die erst für die Darstellung auf dem Bildschirm berechnet werden. Diese Felder können nicht manipuliert werden.
Das gleiche gilt für Felder, die zwar Bestandteil der Datenbank sind, aber dort aus anderen Werten berechnet werden. Für diese kann in der Konfiguration angegeben werden, dass sie nur gelesen, nicht aber verändert werden dürfen.

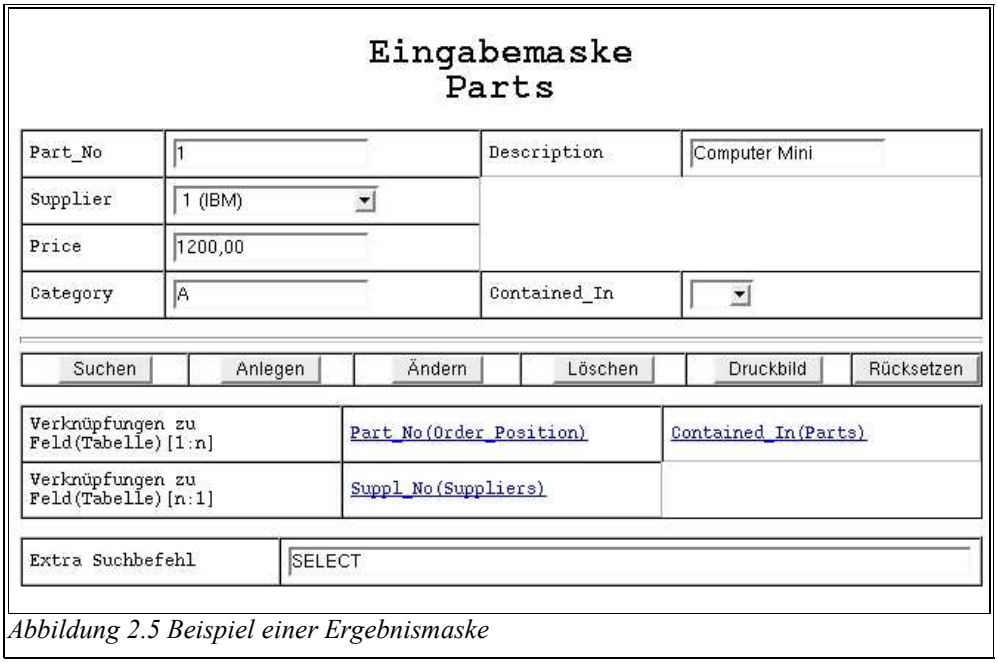


Abbildung 2.5 Beispiel einer Ergebnismaske

2.2.2.2 Die Funktionsknöpfe

In der Suchmaske sind nur 3 unterschiedliche Button zur Auswahl der durchzuführenden Aktion angegeben: 'Suchen', 'Anlegen' und 'Rücksetzen'. In der Ergebnismaske hingegen kommen noch 3 weitere Button hinzu: 'Ändern', 'Löschen' und 'Druckbild'.
Bei dieser Maske liegt ja ein spezifischer Datensatz aus der angewählten Tabelle vor. In den Eingabefeldern können nun Änderungen vorgenommen werden und durch Betätigen des 'Ändern' – Buttons in die Datenbank eingebracht werden, bzw. über den Anlegen – Button als neuer Satz in die Datenbank eingetragen werden. Ein Datensatz kann über den 'Löschen' – Button aus der Datenbank entfernt werden.
Soll ein Datensatz ausgedruckt werden, so kann mit Hilfe des 'Druckbild' – Button eine Web – Seite erstellt werden, die im A4 – Format ausgedruckt werden kann (ohne Datenverlust).
Der 'Rücksetzen' – Button dient dazu, alle in den Eingabefeldern vorgenommenen Eingaben rückgangig zu machen und den Zustand wieder herzustellen ,der vom Server an den Browser übermittelt wurde.

2.2.2.3 Der Querverweisbereich

Der Querverweisbereich ist nur in der Ergebnismaske (s. Bild 2.5) vorhanden, da dieser Bereich Verweise von einem speziellen Datensatz der angewählten Tabelle auf andere Datensätze in anderen Tabellen enthält. Die Inhalte dieses Bereiches werden gesteuert durch die Konfigurationstabelle 'relations', in der die Beziehungen zwischen Tabellen eingetragen werden können. Der Bereich wird auch in der Ergebnismaske nur ausgegeben, sofern Beziehungen zwischen der angewählten und anderen Tabellen bestehen.
Der Bereich ist in zwei Unterbereiche unterteilt:

- Im ersten Bereich werden die Beziehungen zu den Tabellen ausgegeben, die die aktuelle Tabelle referenzieren (die aktuelle Tabelle ist der Vater der anderen Tabellen, bzw. es herrscht eine 1:n Beziehung zwischen den Tabellen).
- Im zweiten Bereich werden die reversen Beziehungen ausgegeben, d.h. Die Beziehungen, bei denen die aktuelle Tabelle die anderen referenziert (die aktuelle Beziehung ist Kind der anderen Tabelle, bzw. es herrscht eine n:1 Beziehung zwischen den Tabellen).

Durch ein Anklicken einer der aufgeführten Beziehungen wird dann ein Suchvorgang in der entsprechenden Tabelle mit dem aktuellen Wert im Beziehungsfeld durchgeführt. Werden Datensätze in der anderen Tabelle gefunden, so werden diese angezeigt wie nach einer Suche über die Suchmaske.

2.2.3 Die Liste von Tabellensätzen

Werden aufgrund einer Suche mehrere Sätze gefunden, so werden sie in einer Liste angezeigt. Wieviel Sätze maximal in einem Fenster angezeigt werden, kann über die Tabelle 'configs' der Konfigurationsdatenbank gesteuert werden. Das Weiterschalten auf die nächsten Sätze oder das Anzeigen aller Sätze erfolgt dann mittels dreier Schalter am Ende der angezeigten Liste.

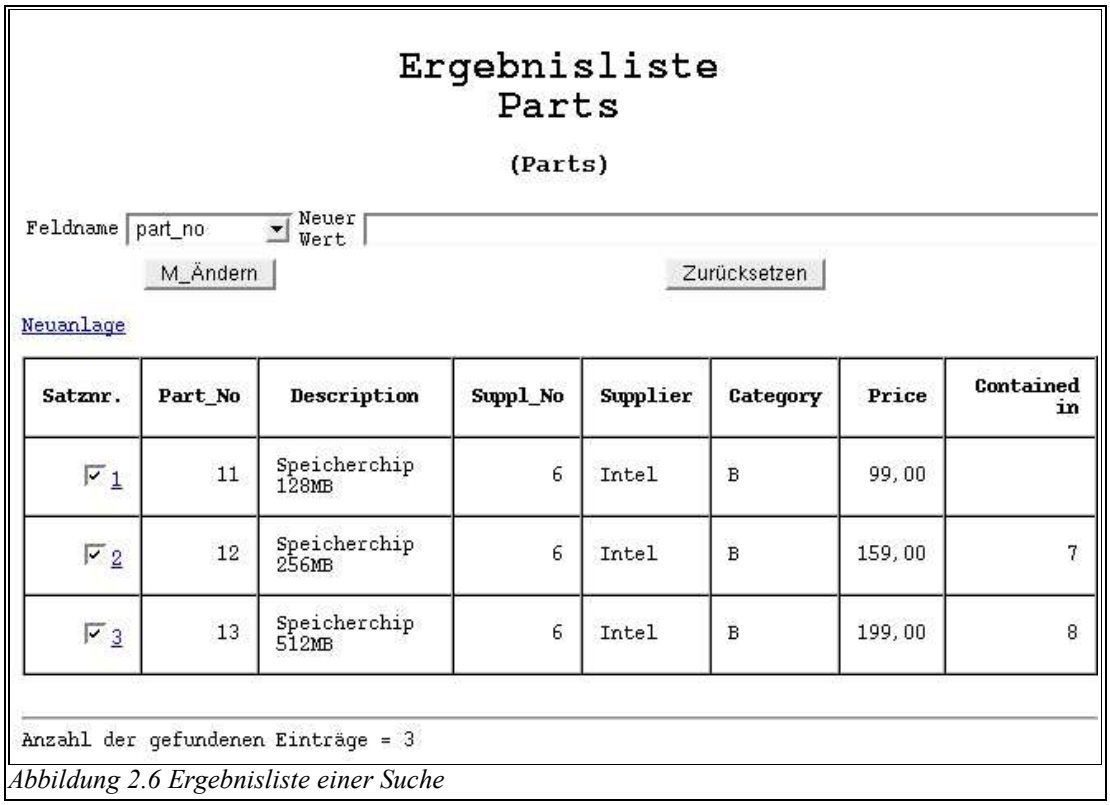


Abbildung 2.6 Ergebnisliste einer Suche

Für das Anzeigen der Sätze selbst gibt es zwei durch die Konfiguration der Tabelle vorgebbare Varianten (s. Abbildung 2.6):

- Mehrere Sätze der Tabelle dürfen auf einmal geändert werden (multiple update) oder
- Nur einzelne angeählte Sätze dürfen geändert weden.

Im ersten Fall sind vor den eigentlichen Sätzen zwei Felder angelegt, in denen das zu ändernde Feld und der neue einzutragende Wert angegeben werden können. Geändert werden alle Sätze, die über die Satznummer-Kontrollkästchen ausgewählt wurden.
Die Liste der Sätze enthält alle ausgewählten Felder. Werden keine konfiguriert, so werden alle Felder angezeigt. Sollen auch berechnete Felder angezeigt werden, so kann dass über die Konfigurationsdatenbank in den Tabellen 'virtual' und 'tabledesign' vorgegeben werden. Auch die Anordnung und Listenüberschrift kann explizit konfiguriert werden.

2.2.4 Aktionen und Berichte

In der Steuerleiste gibt es zwei weitere Bereiche: den Bereich Aktionen und den Bereich Berichte. Diese beiden Bereiche haben Systemintern die gleiche Bedeutung, sie geben nur dem Anwender Hinweise darüber, was sich hinter den angezeigten Möglichkeiten verbirgt.
Ein Bericht oder eine Aktionen für eine Datenbank muss in der Konfigurationsdatenbank als Perl-Funktionen eingetragen sein. Wenn ein Bericht beispielsweise sofort erzeugt werden kann, so kann er auch unmittelbar im Berichtsfenster (s. Abbildung 2.7) ausgegeben werden. Wird die Anzeige des Ergebnisses vollständig in der Perl-Funktion in der Konfigurationsdatenbank gesteuert, so ist für diese Aktion, diesen Bericht, der Wert für das Feld 'ext_result' auf 'true' zu setzen. Dann wird der Rahmen nicht angezeigt (s. Abbildung 2.8).
Es kann aber – dies gilt vor allem für Hintergrundsaktionen – auch geschehen, dass solche Aktionen nur angestoßen werden. Dann erfolgt eine Ausgabe, dass die entsprechende Aktion veranlasst wurde.

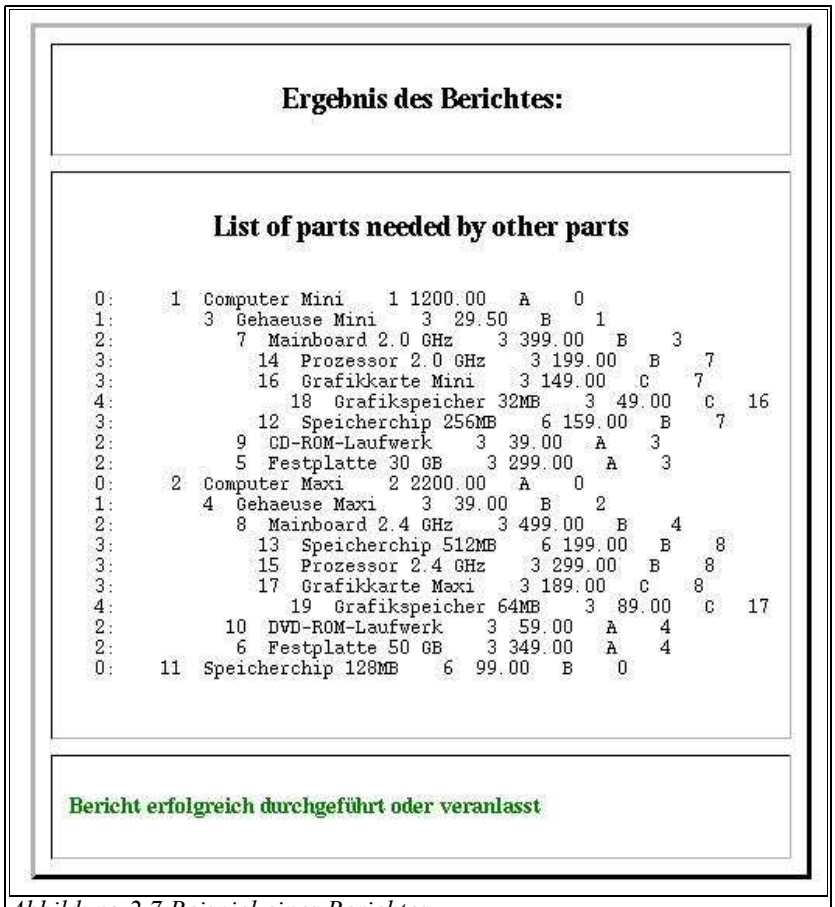


Abbildung 2.7 Beispiel eines Berichtes

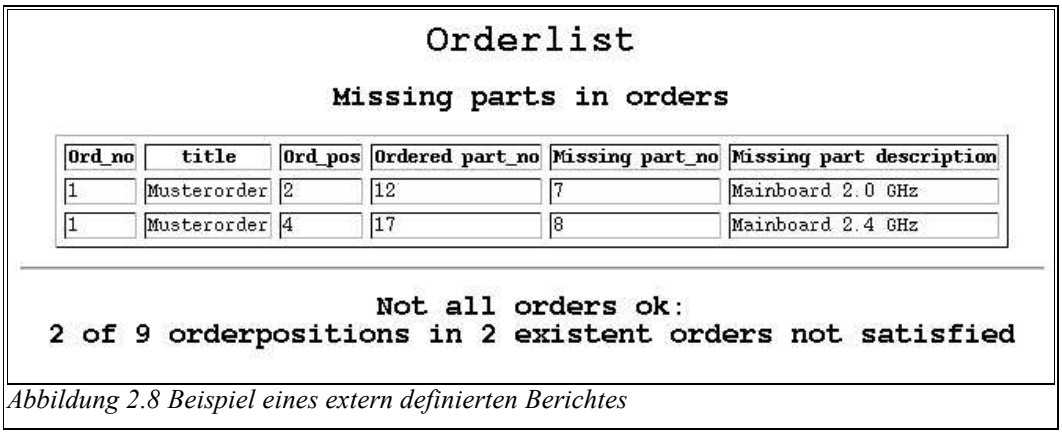


Abbildung 2.8 Beispiel eines extern definierten Berichtes

Neben den datenbankspezifischen Aktionen gibt es auch Standardaktionen (s. Kapitel 3.1.4), die i. a. für alle Datenbanken gelten. In der Konfigurationsdatenbank sind z. Z. 4 Standardaktionen enthalten:

- Die Datenbankverwaltung, sie dient der Erzeugung und Löschung von Datenbanken und darf daher nur vom Administrator benutzt werden.
- Die Schemaverwaltung, sie dient der Erzeugung und Löschung von Schemata in der aktuell angewählten Datenbank und darf daher von dem Administrator und dem Besitzer der Datenbank aufgerufen werden. Ein Beispiel für einen solchen Aufruf zeigt die Abbildung 2.9.

Abbildung 2.9 Standardaktion Schemaverwaltung

- Die Benutzerverwaltung, mit dieser Standardaktion können vom Administrator neue Benutzer angelegt und bereits vorhandene Benutzer verwaltet werden. Jeder Benutzer kann außerdem seine eigenen Daten mit dieser Standardaktion verwalten.

Abbildung 2.10 Maske für die Benutzerverwaltung durch Administratoren

Ist der aktuelle Benutzer kein Administrator, so sieht die Maske entsprechen der Abbildung 2.11 aus. Zum einen ist der Benutzername nicht veränderbar – zu sehen an dem grau unterlegenten Feld für den Namen –, zum anderen fehlen Eingabemöglichkeiten, die nur der Administrator vergeben darf.

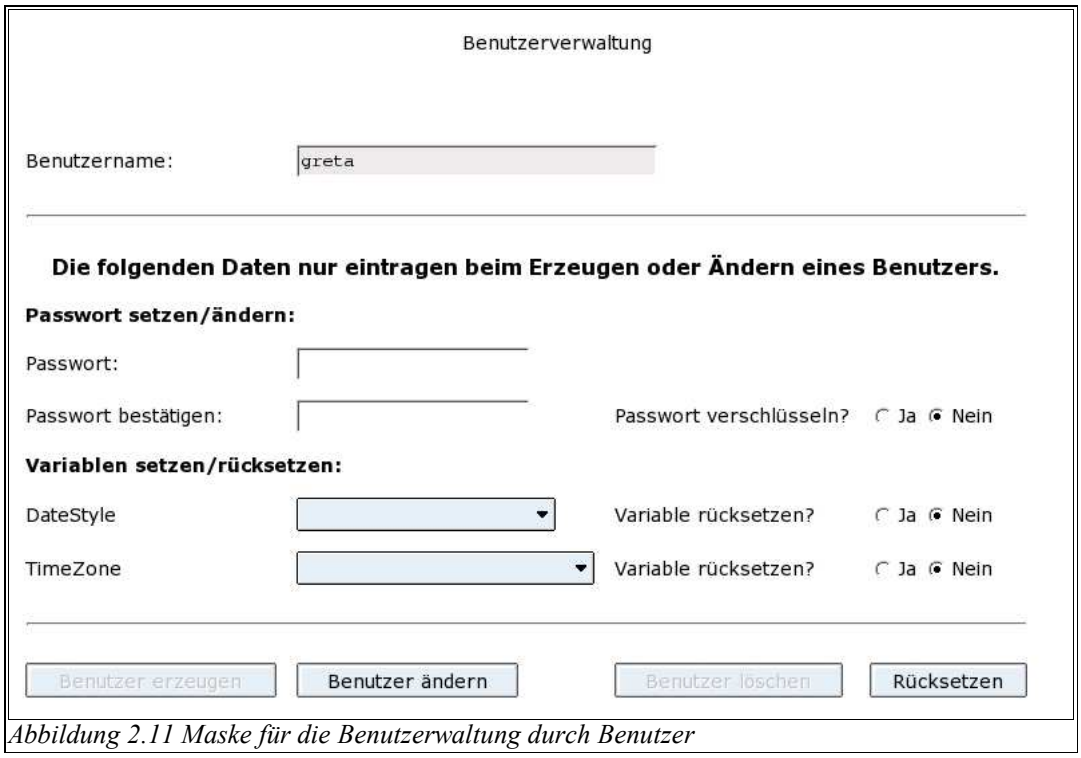


Abbildung 2.11 Maske für die Benutzerwaltung durch Benutzer

- Die SQL-Schnittstelle, mit dieser Standardaktion wird allen zugelassenen Benutzern die generelle SQL-Schnittstelle mit Ausnahme der SQL-Befehle, die durch die anderen drei Standardaktionen abgedeckt werden, freigegeben. Mit der Freigabe sollte vorsichtig umgegenagen werden, aber in jedem Fall kann der Eigentümer der aktuellen Datenbank und der Administrator das Recht bekommen.

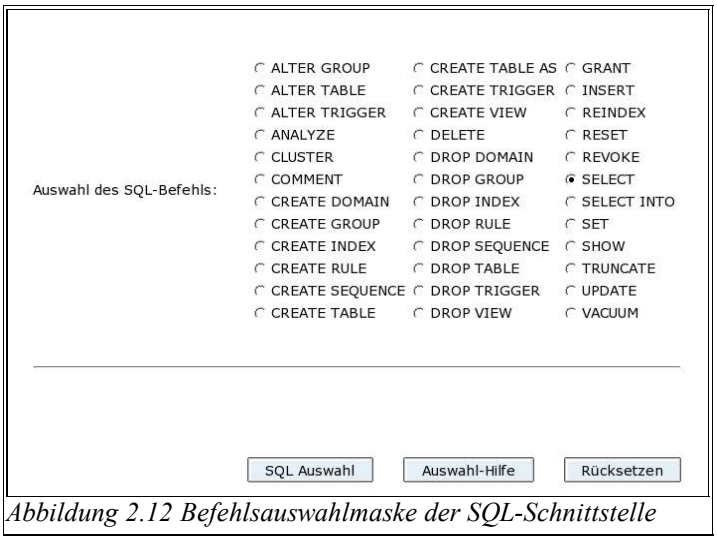


Abbildung 2.12 Befehlsauswahlmaske der SQL-Schnittstelle

Ein Beispiel für die SQL-Schnittstelle zeigt die Abbildung 2.12. Es werden alle SQL-Befehle angezeigt, die ausgewählt werden können und zur Auswahl stehen zwei Buttons zu Verfügung: der erste gibt die Möglichkeit zur Befehlseingabe ohne Unterstützung, der zweite gibt auf der Folgemaske außerdem als Unterstützung die Syntax des Befehls aus. Dennoch sollte der Anwender dieser Schnittstelle mit SQL vertraut sein, denn die Syntax allein gibt natürlich noch keine oder nicht vile Hinweise auf die Semantik eines Befehls.

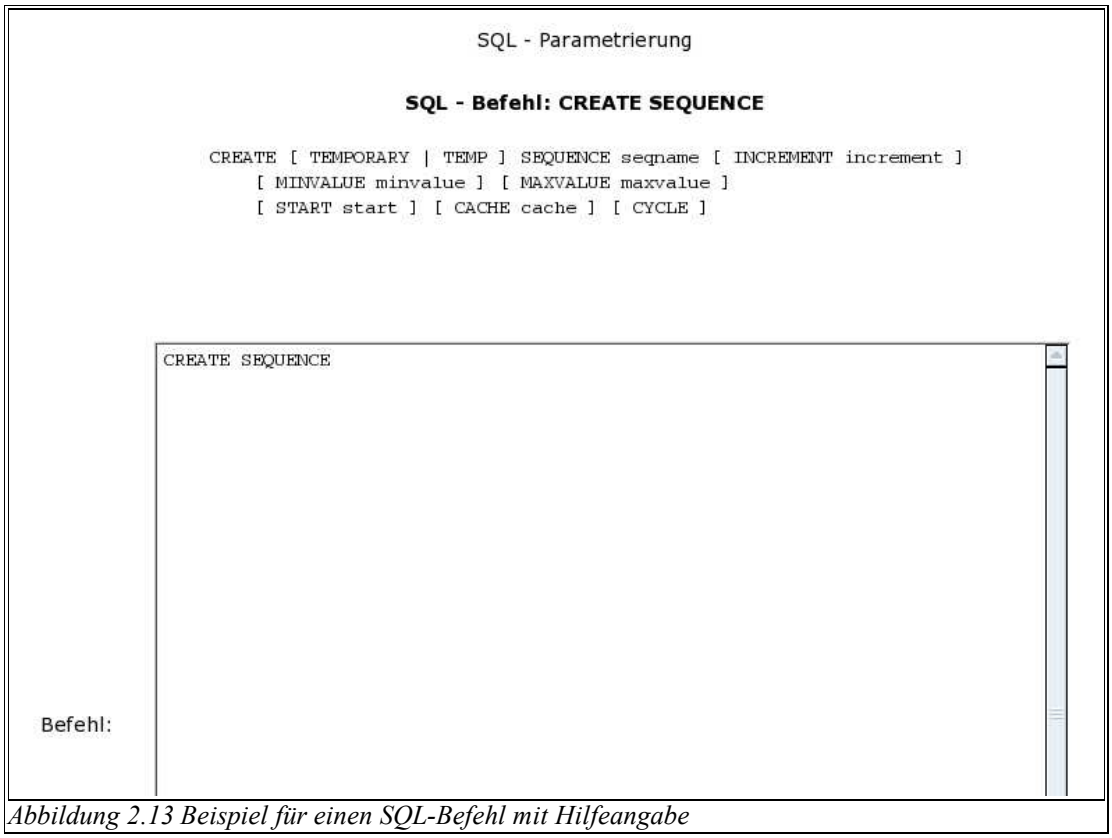


Abbildung 2.13 Beispiel für einen SQL-Befehl mit Hilfeangabe

In der Abbildung 2.13 ist der Befehl 'CREATE SEQUENCE' mit Angabe der Syntax ausgewählt worden. Diese Syntaxbeschreibung kann manchmal sehr umfangreich sein. Sie ist aus den Hilfemodi des postgtreSQL-Befehls 'psql' entnommen.

3 Die Konfiguration einer Datenbank

Zur einfachen Gestaltung der Nutzung bedient sich das Tool selbst der PostgreSQL-Datenbank 'dbengine_info_db', in der alle wichtigen Werte zur Erzeugung von Formularen (Eingabemasken) und Tabellenübersichten bzw. Berichten gespeichert sind. Einige Werte sind auch im Hauptprogramm 'dbengine.pl' gespeichert (als Vorbelegung in dem Hash-Array 'configs'). Dabei handelt es sich aber lediglich um Daten, die benötigt werden, um überhaupt eine Datenbank anzusprechen. Die Tabellen der Konfigurationsdatenbank 'dbengine_info_db' enthalten Konfigurationsvorschriften für jede anzusprechende Datenbank. Doch ist eine Datenbank auch ohne jede weitere Konfigurationsvorschrift bedienbar. Die Konfigurationen gestatten aber eine Definition der Bediener-Oberfläche, der Definition von Zugriffsberechtigungen und noch etliche andere Möglichkeiten.

Tabellenname	Tabellenart	Bedeutung
basedesign	Haupttabelle	In dieser Tabelle werden für jede Tabelle einer Datenbank die wichtigsten Anordnungsdaten abgelegt.
designinfo	Haupttabelle	In dieser Tabelle werden alle Angaben eingetragen, die zur Gestaltung der Benutzeroberfläche für eine Maske zur Dateneingabe, Datenänderung, Datensuche in einer einzelnen Tabelle nötig sind.
tabledesign	Haupttabelle	In dieser Tabelle werden alle Angaben eingetragen, die zur Ausgabe einfacher Listen aus der Tabelle nötig sind. Diese Listen werden bei Suchvorgängen erstellt.
action	Haupttabelle	Diese Tabelle enthält entweder Perlprozeduren oder eine URL zum Durchführen bestimmter Aktionen oder zum Erstellen von Berichten.
relation	Detailtabelle	In dieser Tabelle werden die Beziehungen der Tabellen einer Datenbank untereinander beschrieben. Es wird für jede Tabelle angegeben, ob sie ich auf eine andere bezieht, oder ob eine andere Tabelle sich auf diese bezieht.
virtual	Detailtabelle	Für Masken oder Listen, die berechnete Felder enthalten sollen, werden in dieser Tabelle die Berechnungsvorschriften für die Felder angegeben.
dbengine_user	Detailtabelle	In dieser Tabelle werden die für eine Datenbank zulässigen Benutzer eingetragen.
equation	Detailtabelle	Sollen für eine Datenbank umfangreiche Perlprozeduren mehrfach genutzt werden, so können diese Prozeduren in dieser Tabelle global für die Datenbank vereinbart werden.
databaseconfigs	Wertebereiche	In dieser Tabelle können die systemspezifischen Konfigurationen datanbankspezifisch überschrieben und weitere gesetzt werden.
valuelist	Wertebereiche	In dieser Tabelle stehen Datenbank spezifisch Wertelisten, die für den Maskenaufbau genutzt werden können.
langtexts	Wertebereiche	Diese Tabelle enthält die sprachabhängigen Texte für Dbengine. Diese Texte können somit leicht dem eigenen Bedarf angepasst werden und auch für weitere Sprachen definiert werden.
configs	Wertebereiche	Diese Tabelle enthält die System spezifischen Konfigurationsparameter für Dbengine. Jeder Parameter enthält auch eine Kommentierung, die die eigene Konfiguration erleichtert. Neben den vorhandenen Parametern können weitere definiert werden, die dann in eigenen Prozeduren abgefragt werden können.
helpfile	Wertebereiche	Diese Tabelle enthält sprachspezifische Hilfetexte, die für bestimmte Aktionen vorgegeben werden können, um die Benutzer zu unterstützen.

Tabelle 3.1 Liste der Konfigurationstabellen

Die Tabellen der Konfigurationsdatenbank sind eingeteilt in 'Haupttabellen', 'Detailtabellen' und 'Wertebereiche'. Die Haupttabellen definieren die Schnittstelle zum Benutzer, die Detailtabellen sind dabei als Hilfen notwendig und die Wertebereiche sind Tabellen, die auch unabhängig von der einzelnen Datenbank Wertebereiche für die Nutzung von Dbengine festlegen. Die Tabellen (s. Tabelle 3.1) der Konfigurationsdatenbank werden nachfolgend im Einzelnen beschrieben.

3.1 Die Haupttabellen

3.1.1 Die Tabelle 'basedesign' ('Tabellendefinition')

Diese Tabelle enthält die grundlegenden Angaben für eine Datenbanktabelle. Über die Einträge dieser Tabelle wird das Aussehen der Steuerleiste definiert (s. Kapitel 2.2.1). Folgende Felder sind in der Tabelle 'basedesign' vorhanden:

Feldname	Anzeigename	Bedeutung
'dbasename'	'Datenbank'	Name der Anwender-Datenbank
'tablename'	'Tabelle'	Name der Anwendertabelle
'displayname'	'Anzeigename'	Name der Tabelle in der Anzeige in der Steuerleiste ...
'tableart'	'Tabellenart'	Art der Anwendertabelle. Es gibt dafür drei verschiedene Möglichkeiten: • Haupttabellen (eingetragen als 'maintables') • Detailtabellen (eingetragen als 'detailtables') • Wertebereiche (eingetragen als 'valueranges')
'tableorder'	'Reihenfolge'	Nummer der Tabelle als Reihenfolge der Tabellen in der Tabellenart.
'username'	'Zulässige Benutzer'	Durch Komma getrennte Angabe der zulässigen Benutzer . Mit dem Wort 'ALL' bekommen alle Benutzer Zugriffsrecht.
'description'	'Beschreibung'	Kurze Beschreibung der Aufgabe der Tabelle.

Tabelle 3.2 Die Felder der Tabelle Basedesign

Dbasename	Tablename	Displayname	Tableart	Tableorder	Username	Description
dbengine_info_db	relation	lang(r_relation)	detailtables	1		
dbengine_info_db	virtual	lang(r_virtual)	detailtables	2		
dbengine_info_db	dbengine_user	lang(r_dbengine_user)	detailtables	3		
dbengine_info_db	equation	lang(r_equation)	detailtables	4		
dbengine_info_db	basedesign	lang(r_basedesign)	maintables	1		
dbengine_info_db	designinfo	lang(r_designinfo)	maintables	2		
dbengine_info_db	tabledesign	lang(r_tabledesign)	maintables	3		
dbengine_info_db	action	lang(r_action)	maintables	4		
dbengine_info_db	databaseconfigs	lang(r_databaseconfigs)	valueranges	1		
dbengine_info_db	valuelist	lang(r_valuelist)	valueranges	2		
dbengine_info_db	configs	lang(r_configs)	valueranges	3		
dbengine_info_db	langtexts	lang(r_langtexts)	valueranges	4		
dbengine_info_db	helpfile	lang(helpfile)	valueranges	5		Table for helptexts for actions.

Abbildung 3.1 Liste der Tabellenzuordnung von dbengine_info_db

Sind für eine Datenbank hier keine oder nicht alle Tabellen eingetragen, so werden diese Tabellen alphabetisch in der Steuerleiste unter der Bezeichnung 'Tabellen' angezeigt. Ein Beispiel für die Aufteilung von Tabellen einer Datenbank ist die Konfigurationsdatenbank selbst. Einen Ausschnitt einer Liste für Tabellen-Konfigurationen einer Datenbank, wie sie durch den Suchbefehl

```
SELECT a.* FROM "basedesign" a
        WHERE "dbasename" LIKE 'dbengine_i%'
        ORDER BY dbasename, tableart, tableorder
```

– eingegeben im Feld des Extrasuchbefehls – gefunden wird, zeigt die Abbildung 3.1 für die Zuordnung aller Tabellen der Konfigurationsdatenbank.

3.1.2 Die Tabelle 'designinfo' ('Maskendefiniton')

Diese Tabelle beschreibt den Formularaufbau für eine bestimmte Anwendertabelle. Sie benötigt je Tabellenzeile dabei eine Reihe von Informationen, die in den einzelnen Tabellenfeldern eingetragen werden. Nicht alle Informationen sind nötig, da viele aus den Anwendertabellen-Feldern automatisch ermittelt werden können. Sie sind nur auszufüllen, sofern von den Standardwerten abgewichen werden soll.
Folgende Felder sind in der Tabelle 'designinfo' vorhanden:

Feldname	Anzeigename	Bedeutung
'dbasename'	'Datenbank'	Name der Anwender-Datenbank
'tablename'	'Tabelle'	Name der Anwendertabelle
'fieldname'	'Feldname'	Name des zu beschreibenden Feldes in der Anwendertabelle
'displaystring'	'Anzeigename'	Im Formular auszugebende Feldbezeichnung; wird als Angabe ein Aufruf der Funktion 'lang' eingegeben, so wird der Funktionswert als Index für die Tabelle 'langtexts' genutzt und dort sprachabhängig der eigentliche Anzeigename ermittelt (s. Abbildung 3.2). Fehlt diese Angabe so wird der Feldname ausgegeben, indem der 1. Buchstabe groß geschrieben wird.
'displayinfo'	'Anzeigetyp'	Art der auszugebenden Information im Feld. Weiter unten ist eine genaue Beschreibung der Möglichkeiten dieses Feldes.
'disptable'	'Zusatz 1'	Zusatzfeld für Displayinfo. s. unten bei der Beschreibung von Displayinfo
'dispvfield'	'Zusatz 2'	Zusatzfeld für Displayinfo (s.u.)
'dispdfield'	'Zusatz 3'	Zusatzfeld für Displayinfo (s.u.)
'displayorder'	'Anzeigenummer'	Formularzeile, in der das Feld stehen soll. Dabei wird nur eine größer/kleiner Beziehung berücksichtigt. Eine Reihe ist dabei immer durch die Felder in ihr definiert.
'displaycolumn'	'Anzeigespalte'	Spalte, in der das Feld stehen soll. Dabei werden Felder gezählt, nicht Zeichen und ein Feld besteht immer aus seiner Bezeichnung und dem Wert.
'rowspan'	'Zeilenzahl'	Anzahl der Zeilen, über die sich das Feld erstrecken soll
'colspan'	'Spaltenzahl'	Anzahl der Spalten, die das Feld einnehmen soll
'xevaluation'	'Xevaluation'	Prüf- oder Berechnungsvorschrift (s. Abbildung 3.2) für in das Feld eingegebene Daten: es muss sich dabei um ein mit 'eval' ausführbares Perl-Programm handeln.
'xdefault'	'Xdefault'	Default-Wert für ein Feld. Der Inhalt von 'xdefault' ist ein mit 'eval' zu berechnendes Perl-Programm. Der Wert wird nur berechnet, wenn eine leere Maske ausgegeben werden soll.

Tabelle 3.3 Die Felder der Tabelle Designinfo

Für alle Felder sind in der Datenbank 'dbengine_info_db' bzw. der Testdatenbank Beispiele vorhanden.

Abbildung 3.2 Beispiel einer Designinfo-Maske

Von besonderer Bedeutung ist das Feld 'Displayinfo', da über dieses Feld eine Korrektur der typgebundenen Ausgabe erfolgen kann. Auch Verbindungen zu anderen Tabellen erfolgen über dieses Feld.
Hier daher die Bedeutung der verschiedenen **'displayinfo'** Einträge:

NONE:
: Dieses Feld wird nicht ausgegeben und nicht berechnet.

COMPUTATION:
: Diese Felder werden, wenn ein Wert vorhanden ist ausgegeben, aber es kann kein Wert dafür eingegeben werden. Es handelt sich um berechnete Felder (s. Feld 'price' in Tabelle 'order_position').

SERIAL:
: Dieses Feld wird, wenn Daten vorhanden angezeigt, aber nicht bearbeitet, d.h. es werden keine neuen Daten für 'insert' und 'update' berechnet. Diese Feldart dient zur Ausgabe des PostgreSQL - Datentyps 'serial'.

CURRENCY:
: Sollen Gleitpunktfelder als Währung ausgegeben werden (statt mit dem SQL-gerechteren Format DECIMAL(6,2) bzw. NUMERICAL(6,2)), so kann dies mit dem Eintrag 'CURRENCY' in dem Feld 'displayinfo' geschehen. Dieser Hilfstyp ist veraltet und kommt noch aus einer Zeit, als postgreSQL den Datentyp NUMERIC nicht kannte.

VIRTUAL:
: Felder mit dieser Angabe im Feld 'displayinfo' sind berechnete Felder, die nur ausgegeben werden, aber nie in der Datenbank abgespeichert werden. Diese Felder werden aktuell über das kleine Perl-Programm in 'xequation' (s. Kapitel 3.2.2) berechnet. Zur Berechnung stehen die Werte der anderen Felder dieser Tabellenzeile in '$rvalues' mit den Feldnamen als Keys zur Verfügung. Tritt während der Berechnung ein Fehler auf, der als solcher zurückgemeldet werden kann, so kann dies mit 'die("...")' geschehen, da die Funktion 'eval' dann diesen Fehler zurückmeldet.

popup:
: Für Felder dieser Art wird ein Popup-Menu aus der dbengine_info_db-Datenbank aus der Datei 'valuelist' generiert. Die Auswahl in der Tabelle 'valuelist (s. Kapitel 3.3.39 geschieht über den Datenbanknamen und den Feldnamen (in 'valuelist' das Attribut 'name', während das Attribut 'display' den Popup-Wert enthält.

relationpopup:

Bei diesem Feld wird ein Popup-Fenster erzeugt, aus dem ein Wert ausgegeben werden kann. Der aktuelle Wert steht dabei an erster Stelle in der Liste und wird beim Aufruf auch angezeigt. Für diesen Typ werden in einigen Feldern Zusatzinformationen erwartet:

Feldname	Zusatzinformation
'Zusatz 1'	Tabelle, in der die Werte für das Popup-Feld stehen. In der Regel handelt es sich dabei um Wertebereichs-Dateien.
'Zusatz 2'	In diese Feld wird der Feldname aus der Popup-Tabelle eingetragen, in dem die zulässigen Werte stehen. Soll die Auswahl aus der Tabelle eingeschränkt werden, so können hier zwei weitere Felder (durch Leerzeichen getrennt) eingetragen werden, die das Testfeld und den Testwert in der Tabelle enthalten.
'Zusatz 3'	Falls erforderlich kann in dem Popup-Feld noch eine zusätzliche Beschreibung angezeigt werden, die als Erläuterung für den Wert dienen kann (ist der Wert nur eine Schlüsselzahl, so kann mit diesem Feld z. B. eine zugehörige Bezeichnung ausgegeben werden.

Tabelle 3.4 Die Bedeutung der Zusatzfelder bei 'relationpopup'

Ein Beispiel soll das verdeutlichen. Der Inhalt der Felder sei:

```
'Zusatz 1' = "maler"
'Zusatz 2' = "pa_no ort berlin"
'Zusatz 3' = "name".
```

Bei diesen Werten werden die Werte des Popup-Fensters mit folgendem Kommando ermittelt:

```
SELECT pa_no,name FROM maler WHERE ort='berlin';
```

Das Popup-Fenster selbst hat dann für jeden einzelnen Wert folgendes Aussehen:

```
<Wert von pa_no>(<name>)
```

In der Testdatenbank ist dafür ein Beispiel für die Tabelle 'parts' enthalten.

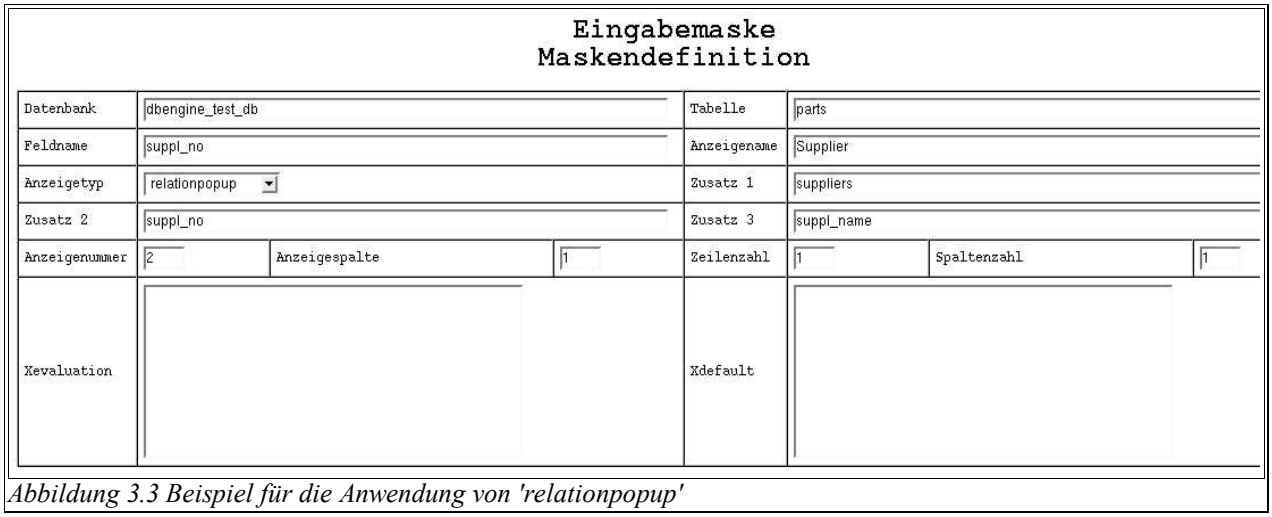


Abbildung 3.3 Beispiel für die Anwendung von 'relationpopup'

Deutlich ist in der Abbildung 3.3 zu sehen, dass die Einträge im Feld 'suppl_no' aus der Tabelle 'suppliers' zu nehmen sind. Zum besseren Erkennen, welcher Wert zu nehmen ist, wird gleichzeitig der Wert aus 'suppl_name' mit angezeigt. Beim Benutzen der Testdatenbank ergibt sich für die Tabelle 'parts' dann eine Maske mit Werten für die 'suppl_no' wie in der Abbildung 2.4 dargestellt.

multiplepopup:

Nur bei 'varchar'- bzw. 'text'-Feldern erlaubt. Dieser Typ bewirkt, dass eine Popup-Liste erzeugt wird, aus der mehrere Werte ausgewählt werden können. Gespeichert werden die Werte in der Datenbank, indem die Einzelwerte durch die Zeichenfolge ';;;' miteinander verbunden werden. Die Auswahl der darzustellenden Listeneinträge geschieht über die Zusatzfelder in gleicher Weise wie bei dem Eintrag 'relationpopup'.

text:

Diese Feldart ist nur dann nötig, wenn Einschränkungen für die Feldlänge und die maximale Datenlänge gemacht werden sollen. Ein Tabellen-Feld des Typs VARCHAR(40) kann z. B. durch Zusatzangaben auf eine maximale Formularfeldlänge von 20 Zeichen und eine maximale Länge von 30 Zeichen eingeschränkt werden. Diese Zusatzangaben stehen in den Feldern (s. Abbildung 3.2):

Feldname	Zusatzinformation
'Zusatz 1'	Länge des Formularfeldes (z. B. 20)
'Zusatz 2'	Maximale Textlänge für die Datei (z. B. 30)

Tabelle 3.5 Die Bedeutung der Zusatzfelder bei 'text'

textarea:

Soll die Eingabe mehrzeiliger Texte in ein Tabellenfeld ermöglicht werden, so muss die Angabe 'textarea' gemacht werden. Als Zusatzangaben werden die Zeilenzahl und Zeichenzahl je Zeile benötigt. Diese Werte stehen in den Zusatzfeldern:

Feldname	Zusatzinformation
'Zusatz 1'	Anzahl der Zeilen des Formularfeldes (z. B. 20)
'Zusatz 2'	Anzahl der Zeichen je Zeile (z. B. 30)
'zusatz 3'	Eintragen des Wertes 'pre', wenn Ausgaben vorformatiert.

Tabelle 3.6 Die Bedeutung der Zusatzfelder 'textarea'

Ein Beispiel für diese Art der Feldbeschreibung ist in der Testdatenbank vorhanden (s. Abbildung 3.4):

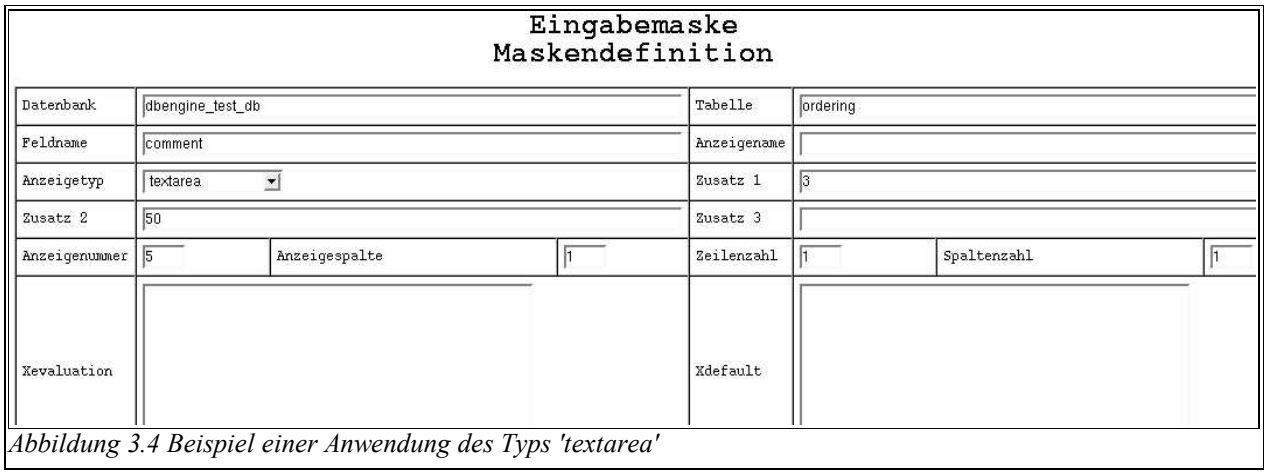


Abbildung 3.4 Beispiel einer Anwendung des Typs 'textarea'

Die Tabelle 'ordering' enthält ein Kommentarfeld 'comment', das mehrzeilige Eingaben verträgt, da ein Kommentar ausführlich sein sollte.
Ist eine Ausgabe nicht vorformatiert, sondern mit HTML-Formatierungen versehen, so können für die Ausgabe Tabellen erzeugt werden, indem die Tabelle eingegrenzt wird mit den

üblichen HTML-Tags '<table>' und '</table>'. Jede Zeile wird dann automatisch erzeugt durch Eingabe eines Zeilenwechsels, Jeder Feldwechsel durch ein Tabulatorzeichen.

Im Übrigen gilt, dass in berechneten Feldern ('xdefault' und 'xevaluation') Variablen-Bereiche und Prozeduren zur Verfügung stehen, wie im Kapitel 3.2.4 beschrieben. Die Werte dieser Variablen können damit jederzeit in den eval-Prozeduren benutzt werden. Wenn für die Prozeduren Eingangsparameter nötig sind, so werden sie stets in der Variablen '$_' zur Verfügung gestellt.

3.1.3 Die Tabelle 'tabledesign' ('Listingdesign')

Diese Tabelle beschreibt den Aufbau eines Listings von mehreren Sätzen aus einer Anwendertabelle. Ein Listing wird aufgerufen durch Suchen über die Suchmaske oder durch die Angabe eines 'Extra Suchbefehl'.
Zur Definition eines solchen Listing sind in der Tabelle 'Tabledesign' folgende Felder definiert:

Feldname	Anzeigename	Bedeutung
'dbasename'	'Datenbank'	Name der Anwender-Datenbank
'tablename'	'Tabelle'	Name der Anwendertabelle
'tablefields'	'Feldliste'	Liste der Felder, die ausgegeben werden sollen. Damit wird gleichzeitig die Reihenfolge der Felder im Listing festgelegt. Ist dieses Feld leer, so werden alle Felder in der Reihenfolge ausgegeben, in der sie von der Datenbank geliefert werden.
'modifyablelist'	'Liste modifizierbar'	Boole'sches Feld, das angibt, ob eine Modifikation eines Feldes für mehrere Sätze zugleich durchgeführt werden darf. In diesem Fall werden in dem Satznummernfeld Kontrollkästchen angelegt (s. Abbildung 2.6). Sätze mit markierten Kontrollkästchen werden geändert, alle anderen nicht. Defaultmäßig sind alle Kontrollkästchen markiert. Im Listing werden außerdem zwei weitere Felder und zwei weitere Knöpfe angelegt, deren Bedeutung in Kapitel 2.2.3 beschrieben ist.
'needsvirtuals'	'Virtuelle Felder'	Boole'sches Feld, das angibt, ob im Listing virtuelle (berechnete) Felder vorkommen. Ist dies der Fall so müssen in der Regel auch die Felder 'xtablestart' und 'xtableitem' ausgefüllt werden. Berechnet werden alle Felder, die für diese Tabelle in der Konfigurationstabelle 'virtual' definiert sind (s. Abbildung 3.9).
'xxquerystring'	'Xxquerystring'	Dieses Feld dient der zusätzlichen Angabe von Suchbedingungen. In der Regel werden hier Sortiervorschriften angegeben. Dies Feld wird mit der Perl-Funktion 'eval' ausgewertet. Es können also umfangreiche Berechnungen erfolgen, sofern nötig. Beim Aufruf der 'eval'-Funktion steht in der Perl-Variablen '$_' der vollständige bis dahin aufgebaute Suchstring (inkl. Der WHERE-Bedingung). Es genügt also eine Stringkonkatenation ('$_ . "ORDER BY <bedingung>") mit der ORDER BY Angabe, um die Sortierung zu veranlassen.
'xtablestart'	'Xtablestart'	Auch dieses Feld wird über die 'eval'-Funktion ausgewertet, und zwar vor der Listenausgabe. In diesem Feld kann daher ein Titel für das Listing und eine Kopfzeile definiert werden, die vor den Zeilen der einzelnen Sätze ausgegeben werden sollen (s. Abbildung 3.5 und Abbildung 3.10). Hier sind die Variablen '$headerline' für die Kopfzeile und '$title' für die Überschrift von Bedeutung.

Feldname	Anzeigename	Bedeutung
'xtableitem'	'Xtableitem'	Über dieses Feld (ebenfalls ein 'eval'-Feld) kann dann die Ausgabe einer einzelnen Satzzeile gesteuert werden. Die Inhalte aller gelesenen und berechneten Felder befinden sich in der Referenz '$rvalues' auf ein Hash-Array, wobei der key der interne Feldname ist. Die Ausgabe eines Satzes wird in die Variable '$line' eingetragen. Ein Beispiel für eine solche satzweise Ausgabe ist in der Abbildung 3.10 zu sehen.
'xtableend'	'Xtableend'	Dieses 'eval'-Feld kann benutzt werden, um am Ende des Listings spezielle Ausgaben zu tätigen. Standardmäßig steht am Ende eines Listings die Anzahl der gefundenen Sätze, doch kann diese Ausgabe durch eine Perl-Prozedur verändert werden. Die Anzahl der gefundenen Sätze steht in der Variablen '$ntuples' zur Verfügung.

Tabelle 3.7 Die Felder der Tabelle Tabledesign

Für die evaluierbaren Felder gilt die gleiche Werteversorgung wie sie in der Tabelle 3.13 und der Tabelle 3.14 beschrieben sind. Zusätzlich sind einige Variable vorhanden, die gesetzt werden können. Die jeweils wesentlichen sind bei der Beschreibung der einzelnen Felder mitbeschrieben. Die Abbildung 3.5 zeigt eine der Standardeinstellungen für die Testdatenbank. Es wird die Reihenfolge der Sätze angegeben, eine Tabellenkopfzeile definiert (der Titel aber nicht geändert) und eine ausgabeprozedur für jede Zeile angegeben. Weitere Beispiele für die Definition von Listings sind sowohl für die Tabellen jder Konfigurationsdatenbank als auch für die Testdatenbank im System vorhanden.

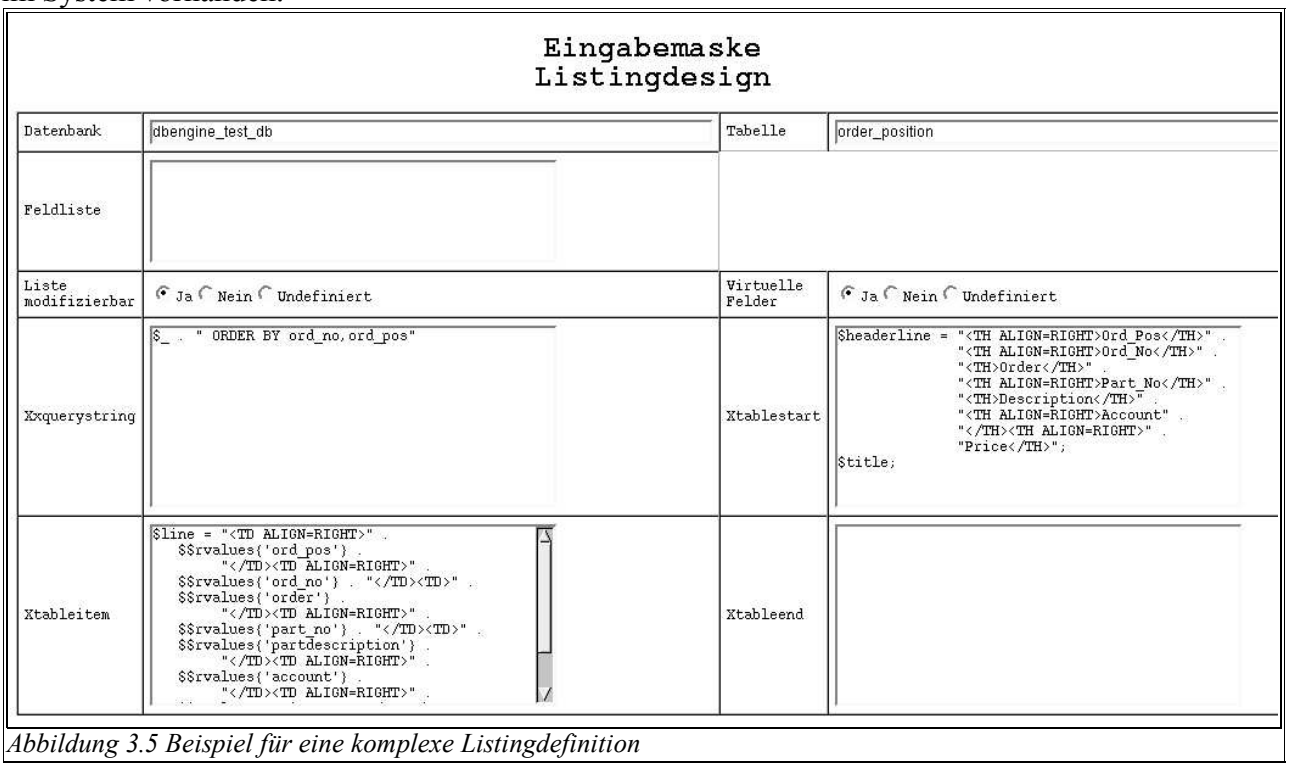


Abbildung 3.5 Beispiel für eine komplexe Listingdefinition

3.1.4 Die Tabelle 'action' ('Aktions-/Berichtsdefinition')

In dieser Tabelle werden die Perl-Prozeduren für die Datenbankspezifischen Aktionen und Berichte definiert. Es werden dabei unterschieden:

- Berichte und

- Aktionen. Bei den letzteren wird weiter unterschieden zwischen
    - Datenbankspezifischen Aktionen und
    - Standardaktionen. Als Standardaktionen sind z.Z. folgende Aktionen definiert:
        - Datenbankverwaltung (__STD_dbcreate): nur für den Administrator aus der angewählten Datenbank 'dbengine_info_db' zulässig.
        - Schemaverwaltung (__STD_schema_manage): ist in allen Datenbanken möglich, aber nur für den Administrator und den Eigentümer der aktuell aufgerufenen Datenbank (s. Abbildung 2.9).
        - Benutzerverwaltung (__STD_manage_user): in in allen Datenbanken möglich und auch für alle Benutzer, bei Benutzern, die nicht gleichzeitig Administratoren sind allerdings nur eingeschränkt: nur für sich selbst darf der Benutzer einige Daten ändern (s. Abilldungen 2.10 und 2.11).
        - SQL-Schnittstelle (__STD_global_SQL): Diese Standardaktion kann für jeden Benutzer freigegeben werden, sofern ihm wenigstens das aktuell angewählte Schema gehört (s. Abbildungen 2.12 und 2.13).

Die Tabelle enthält folgende Felder:

Feldname	Anzeigename	Bedeutung
'dbasename'	'Datenbank'	Name der Anwender-Datenbank. Für Standardaktionen kann hier auch der Wert '__ALL' eingetragen werden: dann ist die Aktion für alle Datenbanken zulässig.
'actionname'	'Aktion'	Name der Aktion oder des Berichtes, unter dem sie auf der Steuerleiste erscheinen sollen. Namen von Standardaktionen beginnen stets mit der Zeichenfolge '__STD_'
'actionorder'	'Reihenfolge'	Reihenfolge der Aktion oder des Berichtes auf der Steuerleiste. Die Nummern größer 1000 sind für Standardaktionen vorbehalten.
'displayname'	'Anzeigename'	Anzeigename der Aktion
'report'	'Bericht'	Bool'sches Feld zur Angabe, ob es sich um einen Bericht oder eine Aktion handelt. Behandelt werden beide Arten intern gleich, der Unterschied besteht in der Anordnung auf der Steuerleiste: es gibt für beide Arten extra Bereiche, damit die Anwender schneller erkennen, um welche Art es sich logisch handelt.
'ext_result'	'Ergebnis extern'	Bool'sches Feld zur Angabe, ob für einen Bericht oder eine Aktion der Erfolg oder Misserfolg angezeigt werden soll, oder ob die durchgeführte Perl-Prozedur dieses selbst durchführt..
'in_list'	'Anzeigen'	Bool'sches Feld zur Angabe, ob diese Aktion bzw. dieser Bericht in der Steuerleiste angezeigt werden soll, oder ob sie/er nur intern beutzt wird. Beispiele für den letzteren Fall sind in der Datenbank 'ereignisse' vorhanden.
'usernames'	'Zulässige Benutzer'	Angabe einer Benutzerliste. Diese Benutzer dürfen die Aktion oder den Bericht starten. Ist die Benutzung für alle zulässig, so kann statt dessen der Wert 'ALL' eingetragen werden. Mit den Namen 'DB_OWNER' bzw. 'SCHEM_OWNER' kann die Erlaubnis gegeben werden für den jeweiligen Datenbankbesitzer bzw. den Eigentümer des aktuell genutzten Schemas in der Datenbank. Nur bei den zugelassenen Benutzern erscheinen die Aktionen bzw. Berichte in der Steuerleiste.
'actiontext'	'Anweisungen'	Angabe der Perl-Prozedur, die beim Start der Aktion oder des Berichtes durchgeführt werden soll.

Tabelle 3.8 Die Felder der Tabelle Action

In der Testdatenbank 'dbengine_test_db' sind drei Beispiele für Berichte bzw. zwei für datenbankspezifische Aktionen enthalten (s. Abbildung 2.3). Die Perl-Prozeduren können sehr umfangreich sein. Um das zu verringern, ist es auch möglich in einer getrennten Tabelle (s. Kapitel 3.2.4) Prozeduren zu definieren und diese in dem Perlcode aufzurufen. Dies ist vor allem dann nützlich, wenn diese Prozeduren von verschiedenen Aktionen aufgerufen werden. Ein Beispiel dafür ist in dem Bericht 'Parts_level_list' zu finden (s. Abbildung 3.6). Das Ergebnis dieses Berichtes ist zu finden in der Abbildung 2.7. Da die auszugebenden Texte nicht direkt mit dem Befehl 'print' ausgegeben werden, sondern als Ergebnis des Codestückes zurückgeliefert werden (der Wert des letzten Statements ist das Ergebnis einer 'eval'-Prozedur, in diesem Fall die Variable '$ergline'), wird der Bericht im umrandeten Fenster ausgegeben.

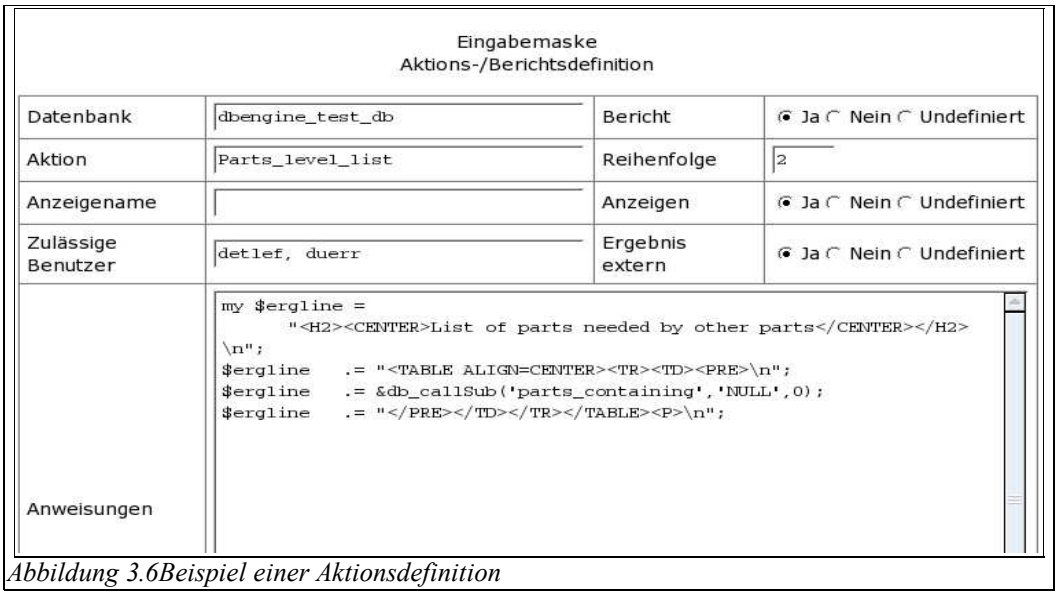


Abbildung 3.6Beispiel einer Aktionsdefinition

Ein anderes Beispiel ist die Aktion 'Check_Order' in der Testdatenbank (s. Abbildung 2.7). In dieser umfangreicheren Aktion werden die wesentlichen Daten direkt ausgegeben und lediglich eine positive bzw. negative Erfolgsmeldung al Rückmeldung aus der 'eval'-Prozedur gemeldet (ein Teil ihres Codes ist zu sehen in dem folgenden Listing).

```
if(@mis_parts) {
   print "<h2><center>Missing parts in orders</center></h2>\n";
   print "<table border cellspacing=5 align=center>\n<tr>" .
         "<th>Ord_no</th><th>title</th>" .
         "<th>Ord_pos</th><th>Ordered part_no</th>" .
         "<th>Missing part_no</th>" .
         "<th>Missing part description</th></tr>\n";
   foreach my $mispart (@mis_parts) {
      my ($ordert, $part_ord, $partd) = split(/;;;/,$mispart,3);
      my ($ord_no, $titel) = split(/:::/,$ordert,2);
      my ($part_no, $ord_pos) = split(/:::/,$part_ord,2);
      my ($partn, $descr) = split(/:::/,$partd,2);
      print "<tr><td>$ord_no</td><td>$titel</td><td>$ord_pos</td>" .
            "<td>$part_no</td><td>$partn</td><td>$descr</td></tr>\n";
   }
   print "</table><P>\n";
   my $missorders = @mis_parts;
   $erg = "Not all orders ok: $missorders of $orderpno orderpositions" .
          " in $orderno existent orders not satisfied";
```

Das Ergebnis des Aufrufs dieser Aktion zeigt (s. Abbildung 2.8) dann eine Tabelle ohne den Ergebnisbericht.
Wie aus den beiden Beispielen zu ersehen ist, erscheint die gerahmte HTML-Tabelle nicht in jedem Fall. Es ist eine ein-spaltige Tabelle. Dabei ist in der zweiten Zeile (unter der Überschrift) entweder der gesamte Bericht (s. Abbildung 2.7) oder nur der Erfolg bzw. Misserfolg (gewollt durch ein gezieltes 'die'-Kommando oder ungewollt durch fehlerhaften Ablauf der angeforderten Perl-Prozedur) zu sehen. Ein Beispiel dafür gibt die Abbildung 3.7.

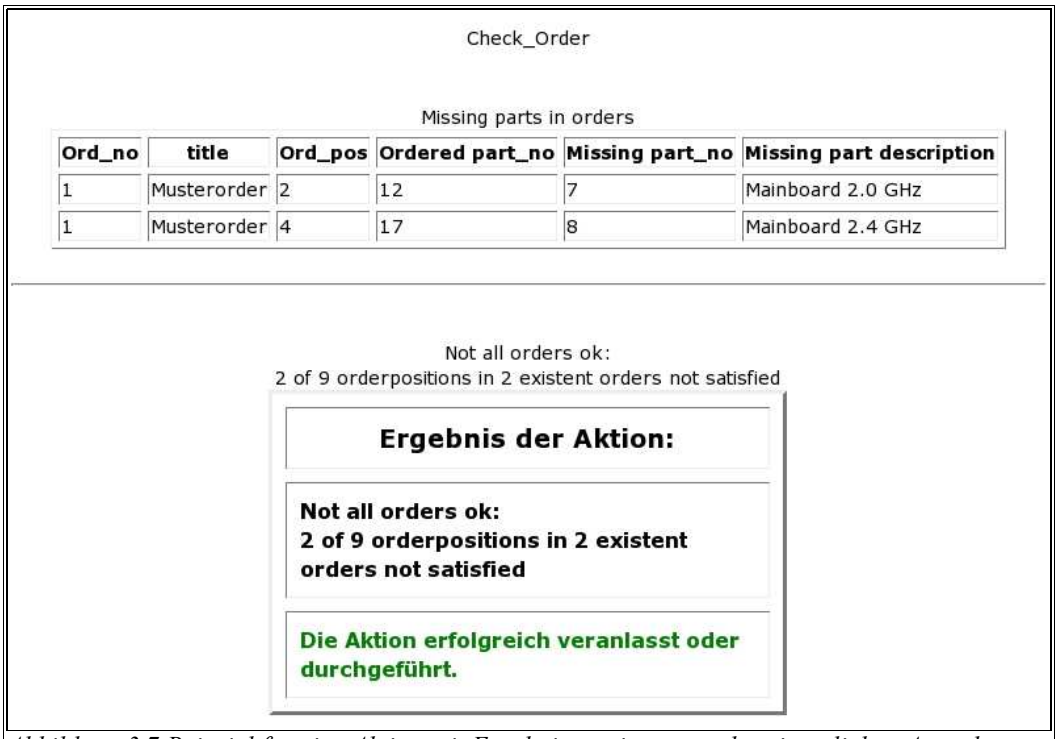


Abbildung 3.7 Beispiel für eine Aktion mit Ergebnisanzeige unter der eigentlichen Ausgabe

Die dritte Möglichkeit ist, dass die Erfolgsmeldung ganz weggelassen wird. Dann muss das Feld 'ext_result' auf 'false' gesetzt werden, wie es für den Bericht 'Orderlist' (s. Abbildung 2.8) geschehen ist.

3.2 Die Detailtabellen

3.2.1 Die Tabelle 'relation' ('Definition von Tabellenbeziehungen')

In dieser Tabelle werden die Beziehungen der Tabellen einer Datenbank untereinander (die Relationen einer relationalen Datenbank) festgehalten. Sie definiert, welche Tabelle welcher anderen über- oder untergeordnet ist. Diese Tabelle enthält folgende Felder:

Feldname	Anzeigename	Bedeutung
'dbasename'	'Datenbank'	Name der Anwender-Datenbank
'parent'	'Elterntabelle'	Name der übergeordneten Tabelle.
'parentfield'	'Feld der Elterntabelle'	Feldname in der übergeordneten Tabelle, die die Relation definiert.

Feldname	Anzeigename	Bedeutung
'child'	'Kindtabelle'	Name der abhängigen Tabelle.
'childfield'	'Feld der Kindtabelle'	Name des Feldes in der abhängigen Tabelle, durch das die Abhängigkeit spezifiziert wird.
'childorder'	'Reihenfolge'	Reihenfolge, in der diese Referenzen in der entsprechenden Ergebnismaske erscheinen sollen.
'childcheck'	'Kindtabelle prüfen'	Bool'sches Feld, das angibt, ob bei UPDATE oder DELETE in der Elterntabelle überprüft werden soll, ob in der Kindtabelle noch Einträge vorhanden sind. In diesem Fall wird die angestossene Aktion nicht durchgeführt.

Tabelle 3.9 Die Felder der Tabelle 'relation'

Die Eintragungen in dieser Tabelle dienen im wesentlichen zur Erzeugung von Querverweisen (s. Kapitel 2.2.2.3). In der Testdatenbank beispielsweise existieren 5 Relationen (s. Abbildung 3.8), die dann auch in den Querverweisen (s. Abbildung 2.5) nach Bedarf wieder auftauchen.

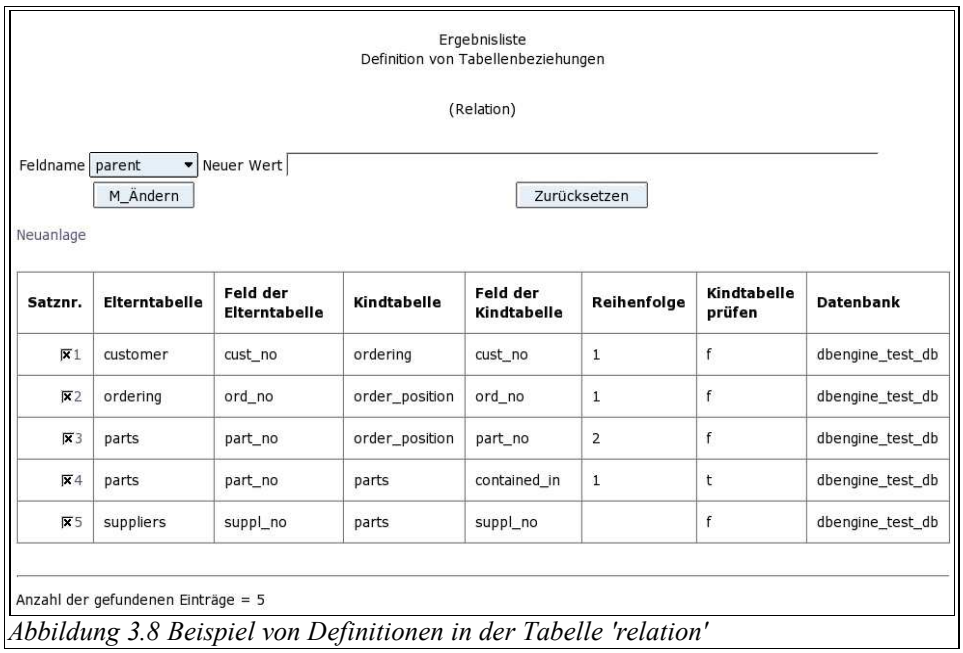


Abbildung 3.8 Beispiel von Definitionen in der Tabelle 'relation'

In dem Eintrag von Satznr. 4 ist für 'childcheck' 't' eingetragen. In diesem Fall wird beim Ändern oder Löschen eines Satzes in der 'parts'-Tabelle geprüft, ob dieses Element Bestandteil eines anderen ist. Ist das der Fall, so wird das Löschen oder Ändern nicht durchgeführt und der Anwender gewarnt, dass erst die anderen, die abhängigen, Sätze geändert werden müssen.

3.2.2 Die Tabelle 'virtual' ('Definition berechneter Felder')

Werden Felder vor der Anzeige berechnet, so wird in dieser Tabelle die Berechnungsvorschrift für diese Felder hinterlegt. Die Tabelle 'virtual' enthält folgende Felder:

Feldname	Anzeigename	Bedeutung
'dbasename'	'Datenbank'	Name der Anwender-Datenbank
'tablename'	'Tabelle'	Name der geforderten Tabelle.

Feldname	Anzeigename	Bedeutung
'fieldname'	'Feldname'	Feldname für das berechnete Feld.
'fieldtype'	'Typ'	Typ des berechneten Feldes.
'level'	'Level'	Level als Ordnungskriterium (Reihenfolge der Felder.
'xequation'	'Xequation'	Berechnungsprozedur für das Feld (Perlcode).

Tabelle 3.10 Die Felder der Tabelle 'virtual'

Die Berechnungsvorschriften gelten sowohl für virtuelle Felder einer Ergebnismaske, die über 'designinfo' (s. Kapitel 3.1.2) konfiguriert werden, als auch für berechnete Felder in einem Listing, die über 'tabledesign' (s. Kapitel 2.2.3) konfiguriert werden. Die Abbildung 3.9 zeigt eine Ergebnismaske für die 'virtual'-Konfiguration eines Feldes in der Testdatenbank.

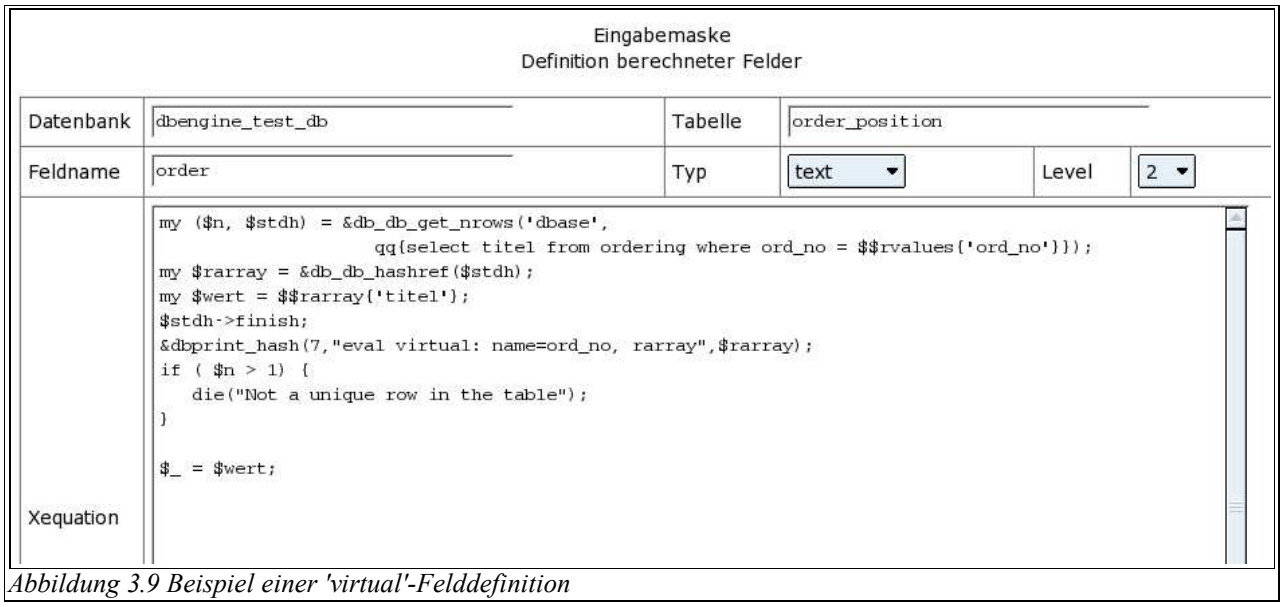


Abbildung 3.9 Beispiel einer 'virtual'-Felddefinition

Mit Hilfe dieser Definition ist im Beispielslisting (Abbildung 3.10) das Feld 'Order' ermittelt worden. Analoges gilt für das Feld 'Description'.

Satznr.	Ord_Pos	Ord_No	Order	Part_No	Description	Account	Price
1	1	1	Musterorder	3	Gehaeuse Mini	1	29,50
2	2	1	Musterorder	12	Speicherchip 256MB	2	318,00
3	3	1	Musterorder	9	CD-ROM-Laufwerk	1	39,00

Abbildung 3.10 Beispiel eines Listings mit berechneten Feldern ('virtual')

In der Testdatenbanktabelle 'Ordering' sind weitere Beispiele zu finden, auch mit einem anderen Zieldatentyp.

3.2.3 Die Tabelle 'dbengine_user' ('Definition zulässiger Datenbankbenutzer')

Diese Tabelle dient der Festlegung, welche Datenbank von welchen Benutzern genutzt werden darf. Nur die Datenbanken, die hier für einen Benutzer gefunden werden, werden in der Steuerleiste (s. Kapitel 2.2.1) aufgeführt. Diese Tabelle besitzt folgende Felder:

Feldname	Anzeigename	Bedeutung
'dbasename'	'Datenbank'	Name der Anwender-Datenbank
'username'	'Zulässige Benutzer'	Liste von Benutzernamen, die diese Datenbank aufrufen dürfen.

Tabelle 3.11 Die Felder der Tabelle 'dbengine_user'

In der Abbildung 3.11 sind die Beispiele für unterschiedliche Einträge vorhanden. Der Eintrag **ALL** im Feld 'username' bedeutet, dass jeder Benutzer diese Datenbank anfragen kann. Es gibt noch einen weiteren Eintrag mit Sonderbedeutung, das ist der Eintrag **Admins**. Dieser Eintrag heißt, dass nur die Administratoren (s. Kapitel 3.3.4) diese Datenbank benutzen dürfen. Das gilt insbesondere für die Konfigurationsdatenbank selbst.

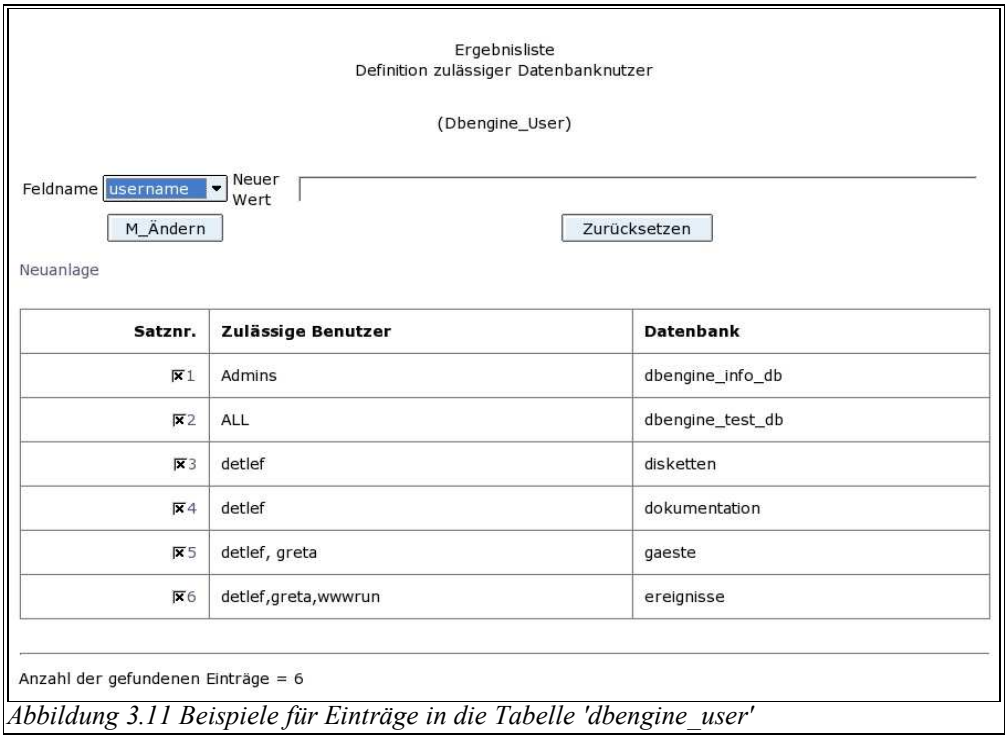


Abbildung 3.11 Beispiele für Einträge in die Tabelle 'dbengine_user'

Die weiteren Einträge in dieser Tabelle sind Beispiele für Benutzernamen, um den üblichen Gebrauch zu kennzeichnen. Wi aus der Tabelle zu sehen ist, können mehrere Namen als Liste, getrennt durch Kommata, angegeben werden. In diesem Fall darf jeder der angegebenen Benutzer die entsprechende datenbank aufrufen. Unabhängig von diesen eintragungen sind natürlich die bei den einzelnen Tabellen einer Datenbank durch das 'GRANT'-Statement angegebenen Benutzereinschränkungen gültig.

3.2.4 Die Tabelle 'equation' ('Definition von Unterprogrammen')

Tritt in einzelnen eval-Prozeduren immer wieder die gleiche Abfolge von Statements auf, so kann diese Folge als Prozedur in der Tabelle 'equation' eingetragen werden. In dieser Tabelle gibt es folgende Felder:

Feldname	Anzeigename	Bedeutung
'dbasename'	'Datenbank'	Name der Anwender-Datenbank
'eqname'	'Bezeichner'	Bezeichner der Perl-Routine, unter dem sie aufgerufen wird.
'content'	'Inhalt'	Perl-Statements

Tabelle 3.12 Die Felder der Tabelle 'equation'

Die Anwendung einer solchen Prozedur geschieht über den Aufruf von '&db_callSub'. Dabei wird als erster Parameter der 'eqname' der Prozedur übergeben. Alle weiteren Parameter stellen dann die Parameter für die Subroutine bzw. Funktion (falls sie einen Wert zurückliefert) dar.
Für die Benutzung in Prozeduren stehen die Variablen, wie sie in der Tabelle 3.13 beschrieben sind, zur Verfügung:

Name der Variablen	Bedeutung
$rfielddesc	Referenz auf ein Hash-Array, mit dem Feldnamen als key, das als Wert ein Hash-Array mit allen notwendigen Angaben enthält. Es sind dies die Daten für das Feld aus der Tabelle und die Beschreibungen des Feldes aus 'designinfo'. Es handelt sich dabei je Feld um folgende Informationen: • TYPE Datentyp des Feldes aus der Datenbanktabelle • DISPLAY Wert von 'displayinfo' • DISPTABLE Wert von 'disptable' • DISPVFIELD Wert von 'dispvfield' • DISPDFIELD Wert von 'dispdfield' • DISPSTR Wert von 'displaystring' bzw. über lang(..) ermittelter Wert • XDEFAULT Perlprozedur für den default-Wert • EVAL Perlprozedur 'xwvaluation' • ROWS Wert von 'rowspan' • COLS echter Wert ermittelt über 'colspan'
$rvalue	Referenz auf ein Hash-Array mit dn Feldnamen als key und den Werten der Felder aus der Datenbank als Inhalt.
$rparms	Referenz auf ein Hash-Array mit allen Parametern, die entweder berechnet oder aus dem Aufruf von 'dbengine.cgi' kommen.
$rconfigs	Referenz auf ein Hash-Array mit allen Konfigurationen aus der Konfigurationstabelle (s. Kapitel 3.3.4)

Tabelle 3.13 Für eine Berechnung mit 'eval' zur Verfügung stehende Werte

Außerdem gibt es die Möglichkeit, viele Prozeduren aufzurufen, die in dem DBENGINE-Tool schon definiert sind. Die wesentlichen Prozeduren sind in der Tabelle 3.14 zusammengefasst.

Prozedurname	Parameter	Ergebnis	Funktion
db_db_get	('dbase', <SELECT-cmnd>, <Bindvalues>)	DB-Handle	Aus der geöffneten Datenbank werden mit dem SELECT-cmd und den SELECT-Parametern (Bindvalues) die gewünschten Sätze gelesen, zurückgeliefert wird ein DB-Handle, über das dann die einzelnen Sätze angefordert werden können. Dies kann z.B. mit der Prozedur '&db_db_hashref(<DB-Handle>' oder mit '<DB-Handle>->fetch' geschehen.

Prozedurname	Parameter	Ergebnis	Funktion
db_db_get_nrows	('dbase', <SELECT-cmd>, Bindvalues)	(Anzahl Zeilen, DB-Handle)	Diese Prozedur arbeitet genauso, wie die Prozedur 'db_db_get. Zusätzlich liefert sie die Anzahl der gefundenen Sätze zurück.
db_db_hashref	(DB-Handle)	Ref-Hash-Array	Diese Prozedur liefert eine Referenz auf ein Hash-Array zurück. Der Key für den Zugriff auf die Werte ist jeweils der im früheren SELECT-Statement definierte Feldname.
db_get_row	(Tabelle, OID)	-	Diese Prozedur liest aus der gewünschten Tabelle den Satz mit der OID aus und schreibt die Werte in das Hash-Array '%values', auf das über die Referenz '$rvalues' zugegriffen werden kann. Der Key ist jeweils der Feldname.
db_callSub	(Prozedur, par1, par2, ...)	Proz-Erg	Über diese Prozedur können in der Tabelle 'equation' definierte Prozeduren mit den Parametern (par1, par2, ...) aufgerufen werden. Sie liefert das Ergebnis der Prozedur zurück.
gtext	(Index)	Langtext	Diese Prozedur liefert zu einem Index den sprachspezifischen Text aus der Tabelle 'langtexts' (s Kapitel 3.3.3). Ausgewählt wird der Text je nach Browsereinstellung beim Anwender und vorhandenen Sprachen für diesen Index.
dbWarnText	(Format, wert1, wert2, ...)	Warnhinweis	Diese Prozedur erzeugt aus dem Format und den Werten (wert1, wert2, ...) einen Warnhinweis, der dann mit dem 'print'-Statement direkt ausgegeben werden kann.
dbWarning	(Format, wert1, wert2, ...)	-	Diese Prozedur erzeugt einen Warnhinweis wie die Prozedur 'dbWarnText' und gibt diesen Warnhinweis sofort aus.
dbError	(Format, wert1, wert2, ...)	-	Diese Prozedur erzeugt eine Fehlermeldung beim Anwender, wobei der erzeugte Text auf die gleiche Weise entsteht, wie bei 'dbWarning'. Doch kehrt die Prozedur nicht zur aufrufenden Stelle zurück, sondern beendet das Programm DBENGINE.

Tabelle 3.14 Tabelle der wichtigsten aufrufbaren Perl-Prozeduren

In der Abbildung 3.12 ist als Beispiel der Ausschnitt einer selbstdefinierten Prozedur zu sehen. Sie wird benutzt in dem Bericht in Abbildung 3.7 in Kapitel 3.1.4. Es ist eine rekursive Prozedur, die sich selbst wieder aufruft. Sie macht ausgiebigen Gebrauch von Datenbankaufrufen und liefert die gesamte Liste, wie sie in Abbildung 2.7 in Kapitel 2.2.4 gezeigt ist.

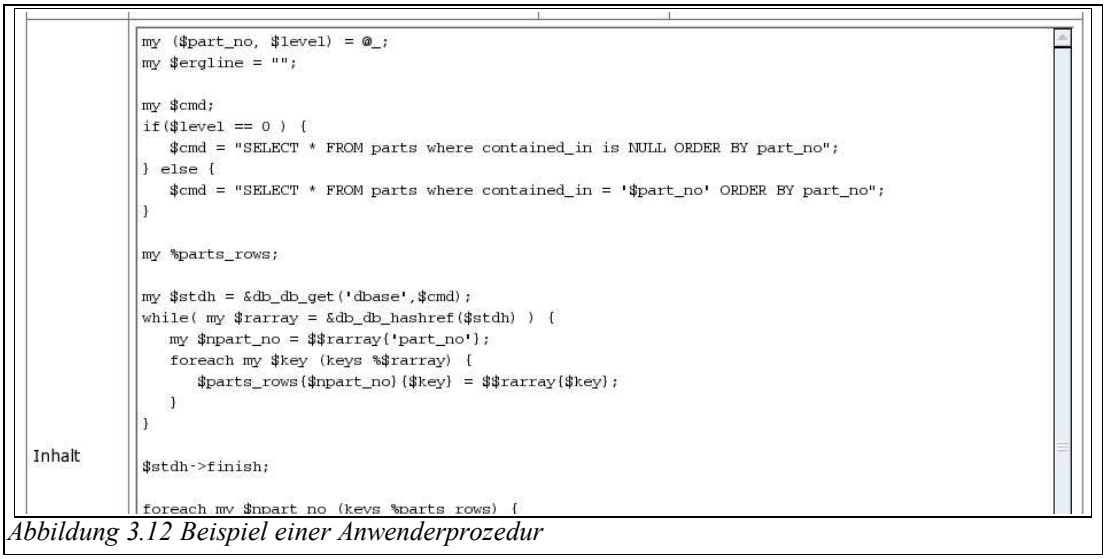


Abbildung 3.12 Beispiel einer Anwenderprozedur

3.3 Die Wertebereichstabellen

3.3.1 Die Tabelle 'databaseconfigs' ('Datenbankspez. Konfigurationen')

Im Kapitel 3.3.4 werden die generellen Konfigurationsmöglichkeiten des Systems beschrieben. Sie gelten für alle datenbanken. Die Tabelle Databaseconfigs hingegen gestattet es diese generellen Konfigurationen für jede einzelne Datenbank speziell zu setzen. Damit werden die generellen Konfigurationen dann für die spezifizierte Datenbank überschrieben. Dazu entält die Tabelle die folgenden Felder:

Feldname	Anzeigename	Bedeutung
'dbasename'	'Datenbank'	Name der Anwender-Datenbank
'configname'	'Name'	Name der Konfigurationsvariablen, für die der Wert dieser Tabelle genutzt werden soll.
'configvalue'	'Wert'	Wert der benutzt werden soll.

Tabelle 3.15 Die Felder der Tabelle 'databaseconfigs'

Im Feld 'Name' wird der Name eingetragen, den die entsprechende Variable gegebenenfalls auch in der Tabelle 'Configs' besitzt. Außer diesen Werten sind aber auch weitere, eigene Konfigurationsvariable eintragbar, die dann in Aktionen, Berichten oder auch sonstigen Listings auswertbar sind. Drei spezielle Werte werden vom System mit ausgenutzt (s. auch Abbildung 3.13):

- 'start_mode':
  Unter diesem Namen kann für den sofortigen Start einer speziellen Seite der angeforderte 'mode' angegeben werden. Möglich sind die Werte 'plain' und 'action'. Bei dem Wert 'plain' kann sofort eine Tabelle und ein Wert daraus ausgewählt werden, deren Inhalt angezeigt werden soll; unter dem Wert 'action' kann eine Aktion oder ein Bericht ausgewählt werden, die/der beim Start sofort auszuführen ist.
- 'start_fname':
  Der Wert in dieser Konfigurationsvariablen bestimmt den Namen eines Parameters, der in der URL zum Aufruf der Startseite gesetzt werden soll.
- 'start_fvalue':
  Diese Konfigurationsvariable definiert den Wert des Parameters, der unter 'start_fname' definiert wurde.

Satznr.	Datenbank	Name	Wert
1	dbengine_info_db	single_sentence	false
2	dbengine_test_db	single_sentence	false
3	ereignisse	single_sentence	true
4	dbengine_test2_db	comment	Test Datenbank zu Behandlung von Schemata
5	ereignisse	start_mode	plain
6	ereignisse	start_fname	table
7	ereignisse	start_fvalue	ereignis
8	ereignisse	std_actions	detlef:__STD_global_SQL
9	dbengine_test2_db	std_actions	detlef:__STD_global_SQL,__STD_global_SQL_do,__STD_global_SQL_execute
10	ereignisse	debug	

Abbildung 3.13 Einige Einträge aus der Tabelle 'Databaseconfigs'

In dem Listing in Abbildung 3.13 ist ein Beispiel für die Startvariablen angeführt. Die Datenbank 'ereignisse' startet beim Aufruf stets mit der Tabelle 'ereignis'. Dies könnte aber durch eine entsprechende Änderung umgesetzt werden auf die Aktion 'Ereignis suchen'. Ein Listing dieser Aktion ist in der Tabelle 'Actions' zu finden. In dem Listing sind außerdem Beispiele für die Änderung der globalen Konfigurationsvariablen 'single_sentence'. In den Datenbanken 'dbengine_info_db' und 'dbengine_test_db' sollen Suchergebnisse, bei denen nur ein Satz gefunden wurde sofort in der 'plain_mode'-Darstellung ausgegeben werden, während bei der Datenbank 'ereignisse' auch bei nur einem Satz die Listingausgabe bevorzugt wird.

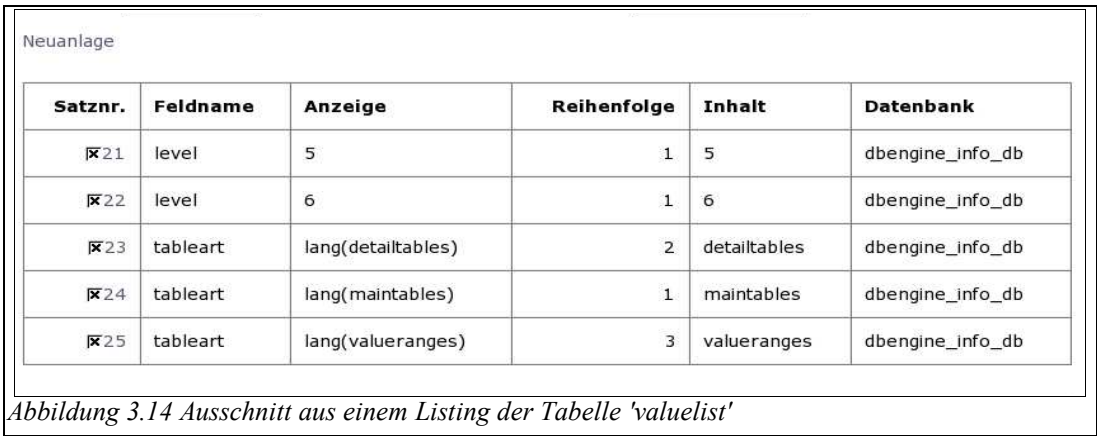
3.3.2 Die Tabelle 'valuelist' ('Wertebereichsdefinition')

Im Kapitel 3.1.2 wird für den Fall eines 'popup'-Feldes schon auf diese Tabelle hingewiesen. Sie enthält die Werte für eine Auswahl. Zur Definition der Werte enthält die Tabelle folgende Felder:

Feldname	Anzeigename	Bedeutung
'dbasename'	'Datenbank'	Name der Anwender-Datenbank
'name'	'Feldname'	Name des Popup-Feldes für das dieser Wert der Tabelle genutzt werden soll.
'content'	'Inhalt'	Wert der benutzt werden soll.
'display'	'Anzeige'	Wert der im Popup-Fenster angezeigt werden soll, um dann für die Datenbank den Wert aus 'content' bereitzustellen. Für diesen Wert gilt das gleiche, wie schon in der Tabelle 'designinfo' (s. Kapitel 3.1.2) für das Feld 'displaystring' (s. Tabelle 3.3), wenn ein Anzeigewert als Funktion 'lang(...)' angegeben ist.
'orderno'	Reihenfolge	Dieser Wert dient der Sortierung der Werte im Popup-Fenster, um so wichtige Werte an den Anfang stellen zu können.

Tabelle 3.16 Die Felder der Tabelle 'valuelist'

Die Abbildung 3.14 zeigt einen Ausschnitt aus einem Listing der Tabelle 'valuelist'. Es sind im wesentlichen die Werte für das Feld 'tableart' der Tabelle 'basedesign' (s. Kapitel 3.1.1). Hier ist zu sehen, dass der Anzeigewert sich sprachspezifisch von dem Wert, der zum Eintrag in die Datenbank ausgewählt wird unterschieden ist.



Neuanlage

Satznr.	Feldname	Anzeige	Reihenfolge	Inhalt	Datenbank
21	level	5	1	5	dbengine_info_db
22	level	6	1	6	dbengine_info_db
23	tableart	lang(detailtables)	2	detailtables	dbengine_info_db
24	tableart	lang(maintables)	1	maintables	dbengine_info_db
25	tableart	lang(valueranges)	3	valueranges	dbengine_info_db

Abbildung 3.14 Ausschnitt aus einem Listing der Tabelle 'valuelist'

Die Abbildung zeigt außerdem, dass das gesamtlisting mehr als 1 Bildschirm ist, weshalb am Ende der Abbildung die Button zur Auswahl der Bildschirmseite angegeben sind.

3.3.3 Die Tabelle 'langtexts' ('Sprachspezifische Texte')

In dieser Tabelle werden die sprachbezogenen Texte hinterlegt. Es wird dabei unterschieden in button-Texte und alle übrigen Texte wie Hinweise, Überschriften usw. Gesucht wird stets über einen Index, der zusammen mit der Sprache den primären Schlüssel der Tabelle bildet. Die Tabelle enthält folgende Felder:

Feldname	Anzeigename	Bedeutung
'indextext'	'Index'	Textueller Index, über den der sprachbezogene Text gesucht wird.
'translate_text'	'Ausgabetext'	Text, der für den Index ausgegeben wird.
'button'	'Button'	Bool'sches Feld, gibt an, ob der Text für einen Button benötigt wird, oder ob es sich um einen Standardtext handelt.
'language'	'Sprache'	Sprache, für die der Ausgabetext gilt.

Tabelle 3.17 Die Felder der Tabelle 'langtexts'

Das Feld 'language' (s. Tabelle 3.17) enthält die Kennzeichnungen für die Sprache in der Form, wie sie auch in den Browsern angegeben werden. Also steht 'de' für 'Deutsch', 'en für 'English' usw. (s. Abbildung 3.15).
Das Feld 'translate_text' (s. Tabelle 3.17) kann in der Form eines Formates für die Perl-Prozedur 'print' angegeben werden. Beim Aufruf der Prozedur 'gtext' (s. Tabelle 3.14) müssen dann die im format benötigten Werte als weitere Parameter mitgegeben werden.
Die Eingabe- oder Ergebnismaske für die Tabelle 'langtexts' ist in der Abbildung 3.15 gezeigt.

Abbildung 3.15 Beispiel einer sprachspezifischen Textdefinition

3.3.4 Die Tabelle 'configs' ('Konfigurationswerte')

In der Tabelle 'configs' werden alle Konfigurationsabhängigkeiten hinterlegt. Sie hat folgenden Aufbau:

Feldname	Anzeigename	Bedeutung
'index'	'Index'	Textueller Index, über den der sprachbezogene Text gesucht wird und unter dem er in dem Hash-Array '%configs' abgelegt wird. Zugreifbar sind die Werte dann in eigenen Prozeduren über die Referenz auf das hash-Array '$rconfigs' (s. Tabelle 3.13)
'values'	'Wert'	Konfigurationsparameter, der über den Index gefunden wird.
'comment'	'Bemerkung'	Beschreibung des Konfigurationsparameters, dient nur der besseren Handhabung.

Tabelle 3.18 Die Felder der Tabelle 'configs'

Mit Hilfe dieser Tabelle können also beliebige aber Datenbankunabhängige Konfigurationen vorgegeben werden. Diese Tabelle kann jederzeit systemspezifisch erweitert werden.

3.3.5 Die Tabelle 'helpfile' ('Hilfetexte')

Diese Tabelle dient der Bereitstellung von Hilfetexten für Aktionen. Ihre Anwendung wird aus einer Aktion mit dem Aufruf '&action_get_help' aufgerufen. Diese Prozedur hat 2 Parameter:

- Den Aktionsnamen und
- den Namen der Hilfegruppe (des Hilfetextes).

Ein Beispiel eines solchen Aufrufes findet sich in der Standardaktion zum Anwenden der allgemeinen SQL-Schnittstelle (s. Abbildung 3.16).

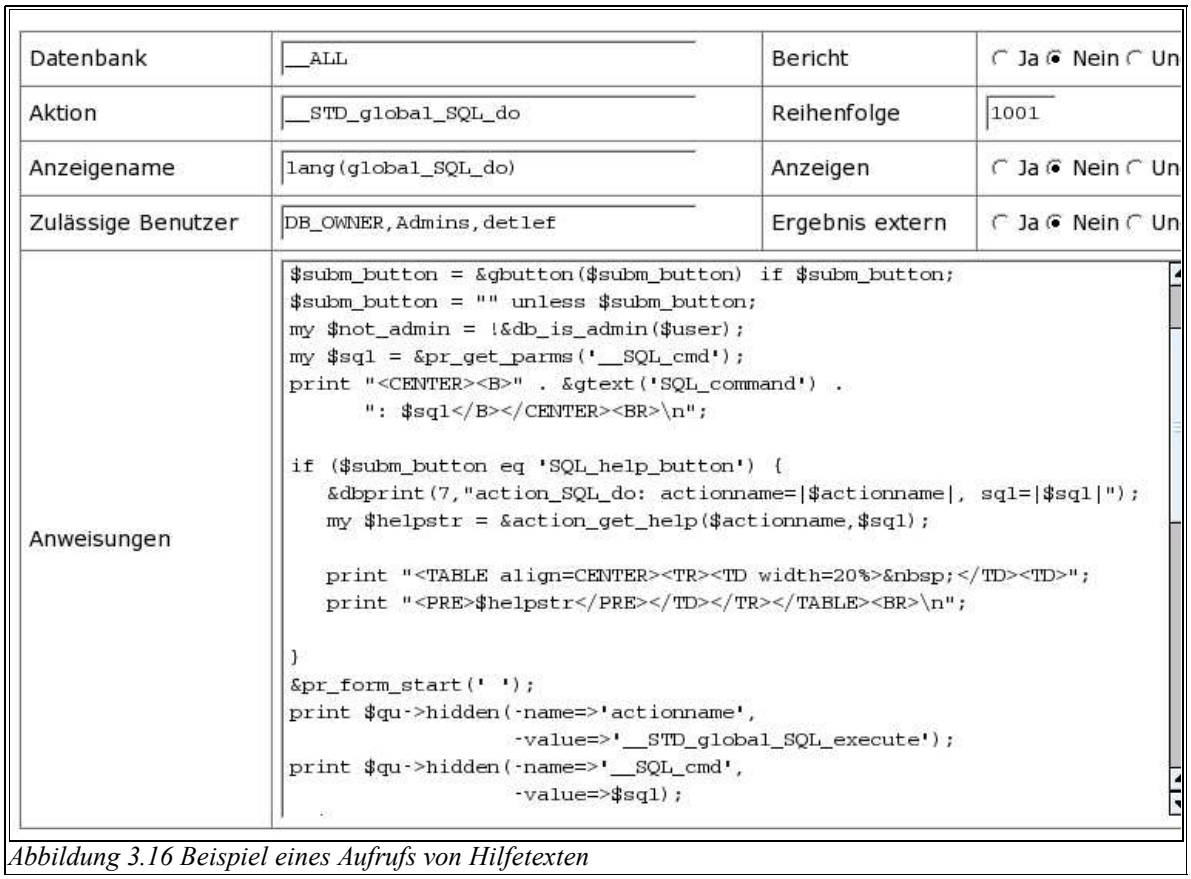


Abbildung 3.16 Beispiel eines Aufrufs von Hilfetexten

Die Tabelle 'helpfile' besitzt zur Festlegung der Hilfetexte folgende Felder:

Feldname	Anzeigename	Bedeutung
'dbasename'	'Datenbank'	Name der Datenbank, für die dieser Hilfetext gilt.
'actionname'	'Aktion'	Name der Aktion, für die dieser Hilfetext gilt.
'helpname'	'Hilfegruppe'	Name der Hilfegruppe, .für die dieser Hilfetext gilt.
'languages'	'Sprache'	Angabe der Sprache, für die dieser Hilfetext gilt.
'helptext'	'Hilfetext'	Angabe eines vorformatierten Hilfetextes. Die Ausgabe dieses Textes erfolgt mit den HTML-Tags '<pre>' und '</pre>'.

Tabelle 3.19 Die Felder der Tabelle 'helpfile'

Im Aufruf 'my $helpstr = &action_get_help($actionname,$sql);' in Abbildung 3.16 werden nur zwei Parameter – wie oben angegeben – mitgegeben, denn die Datenbank ist die aktuell geöffnete und die Sprache wird aus der aktuellen Browsereinstellung entnommen.

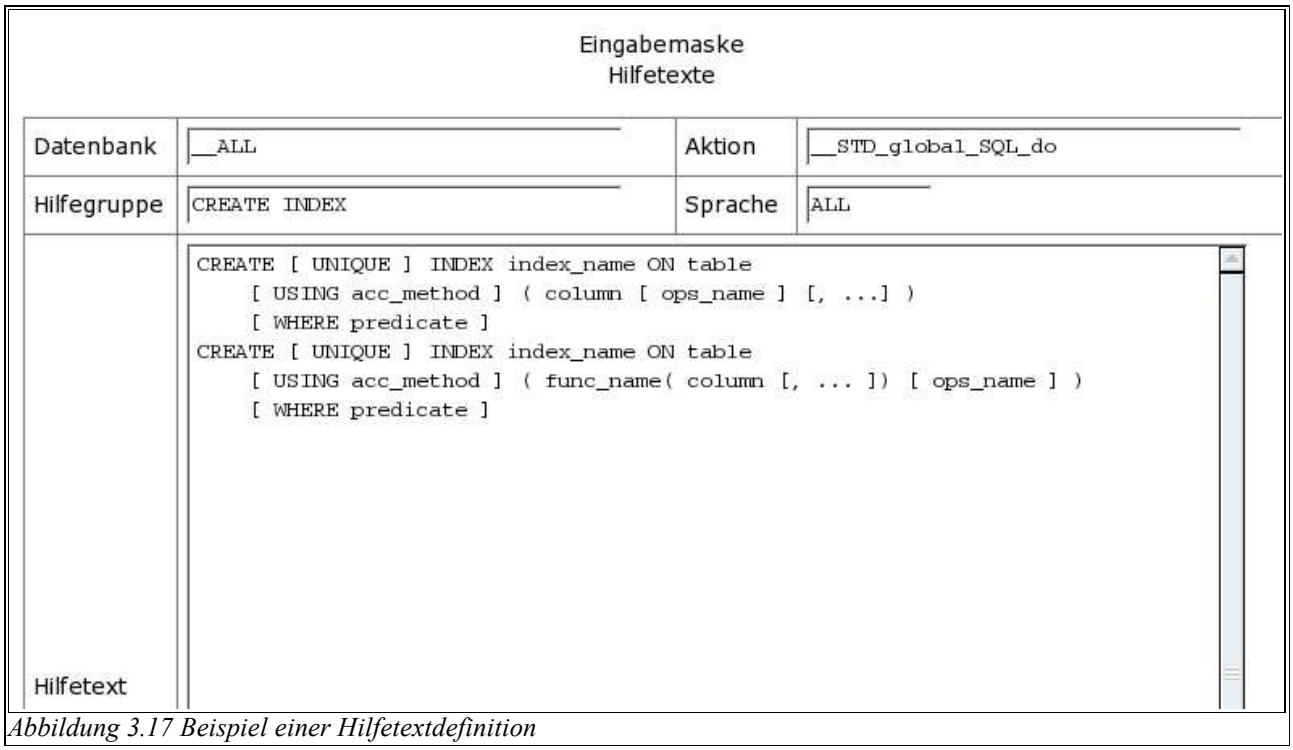


Abbildung 3.17 Beispiel einer Hilfetextdefinition

Ein Beispiel für die Definition eines Hilfetextes ist in der Abbildung 3.17 zu sehen. An dieser Definition kann man erkennen, dass als Sprache auch das Wort 'ALL' eingegeben werden kann, sofern dieser Hilfetext sprachunabhängig ist.
Ein Beispiel für die Anzeige eines Hilfetextes ist in der Abbildung 2.13 schon gezeigt.