Datenbankmodell „Virtuelles Fahrzeug“ Version 1.0

Michael Bellair

gesetzt mit TEX/LATEX

Ausgabedatum: 2007-09-21

Zusammenfassung

Das Datenbankmodell „Virtuelles Fahrzeug“ wird beschrieben. Dieses Dokument ist der GNU Free Documentation License (GNU-FDL) unterstellt und somit im Rahmen dieser Lizenz frei veränderbar. Jede veränderte Version ist jedoch selbst unter der GNU-FDL zu veröffentlichen.
Die Lizenz kann Anhang A entnommen werden. Eine deutsche Übersetzung ist unter http://www.giese-online.de/gnufdl-de.html zu finden.

Inhaltsverzeichnis

1 Einleitung

Es wird die Struktur des Datenbankmodells vorgestellt. Jedem Datenbankschema ist ein Abschnitt zugeordnet. Ziel und Zweck des Schemas werden beschrieben.

2 Das Datenbankmodell

Innerhalb eines Schemas können die Tabellen aufeinander referenzieren (Fremdschlüssel), wobei keine Kreisreferenz (Zirkelbezug) angewendet wird.
Unter dieser Vorraussetzung sind stets Tabellen vorhanden

- auf die referenziert wird und die selbst **nicht** auf eine weitere Tabelle verweisen (Referenztabellen),
- auf die **nicht** referenziert wird und die selbst auf Referenztabellen verweisen (Konsumtabellen),
- auf die referenziert wird und die selbst auf Referenztabellen verweisen (RefKonTabellen) und
- auf die **nicht** referenziert wird und die selbst **nicht** auf Referenztabellen verweisen (Trivialtabellen).

Zur Darstellung der Datenbankstruktur (Abhängigkeiten der Tabellen) werden Konsumtabellen am rechten Rand (rechte Spalte), Referenztabellen am linken Rand (linke Spalte) und RefKonTabellen in den dazwischenliegenden Spalten angeordnet.
Sollte eine RefKonTabelle aus mehreren Spalten referenziert werden, so wird sie so weit wie möglich nach Links verschoben.

Die Sortierfolge der Tabellen in einer Spalte erfolgt lexikalisch.
Die Sortierfolge der Spalten einer Tabelle erfolgt ebenfalls lexikalisch, ausgenommen die Spalten `id`, `idfrei` und `notizen`.
Die Spalte `id` ist stets die erste, `idfrei` die zweite und `notizen` die dritte Spalte einer Tabelle.

Als Datenbankmanagementsystem kommt PostgreSQL zum Einsatz.
Unter der Vorraussetzung, daß der Benutzer `micha` angelegt ist, kann die Datenbank `vf` vom Benutzer `micha (-U)` für den Eigentümer `micha (-O)` mit der Zeichensatzcodierung `MULE_INTERNAL` mittels Postgres-Befehl

```
createdb -U micha -O micha -E MULE\_INTERNAL vf 'Datenbank des Virtuellen Fahrzeugs'
```

angelegt werden.
Die aktuell gültigen Versionen der Datenbank lauten

```
createdb -U micha -O micha -E MULE\_INTERNAL vf\_1\_0 'Freigegebene, nicht mehr veränd
createdb -U micha -O micha -E MULE\_INTERNAL vf\_1\_1 'Aktuelle Entwicklerversion der
```

Innerhalb des Datenbankmodells gelten die folgenden allgemeinen Konzepte:

- Aus dem Schema `intern` darf auf keine Tabellen eines anderen Schemas referenziert werden. Auf das Schema `prj` darf aus keinem anderen Schema referenziert werden.
- Alle Spalten (neben `id` und `idfrei` aus `intern.alle`) welche mittels Fremdschlüssel auf andere Tabellen verweisen, müssen mit einem Vorgabewert (`DEFAULT 0`) versehen und dürfen nicht leer (`NOT NULL`) sein. Somit können Referenztabellen in der `SELECT - WHERE` - Anweisung mittels `AND` ausgelesen werden.

- Alle anderen Spalten dürfen keinen Vorgabewert haben und dürfen leer sein. Auf diese Weise können mittels `COALESCE` leere Werte entfernt werden
- Kommentare werden im Klartext eingegeben. Sie können mittels beliebiger TEX/LATEX-Schlüsselworte formatiert werden.
- Plausibilitätsprüfungen werden innerhalb der Funktionen durchgeführt. Um die Geschwindigkeit der Funktionen möglicht groß zu halten, werden Standardprüfungen (z.B. Einhalten von Fremdschlüsselbeziehungen, Typumwandlungen) von Postgres durchgeführt (derartige Fehlermeldungen unterscheiden sich sehr von den selbst implementierten Fehlermeldungen). Es gilt das Prinzip: Prüfungen, welche von Postgres ausgeführt werden müssen nicht nocheinmal implementiert werden !
- Die Funktionsparameter (einschl. Rückgabewerte) werden vom Datentyp `varchar` oder `varchar[]` ausgeführt.

Funktionsnamen werden wie folgt festgelegt:

- `aktualisiere__...` — Werte aktualisieren (`UPDATE`)
- `erzeuge__...` — ein Ergebnis mittels komplexer Operationen erzeugen
- `gib__...` — Einzelwerte oder Liste zurückgeben (`SELECT`)
- `ist__...` — Werte prüfen (Aspekt, Forderung, ... erfüllt ?)
- `loesche__...` — Werte löschen (`DELETE`)
- `nimm__...` — Werte übernehmen (`INSERT`)
- `pruefe__...` — Werte anhand vorgegebener Randbedingungen auf Plausibilität prüfen

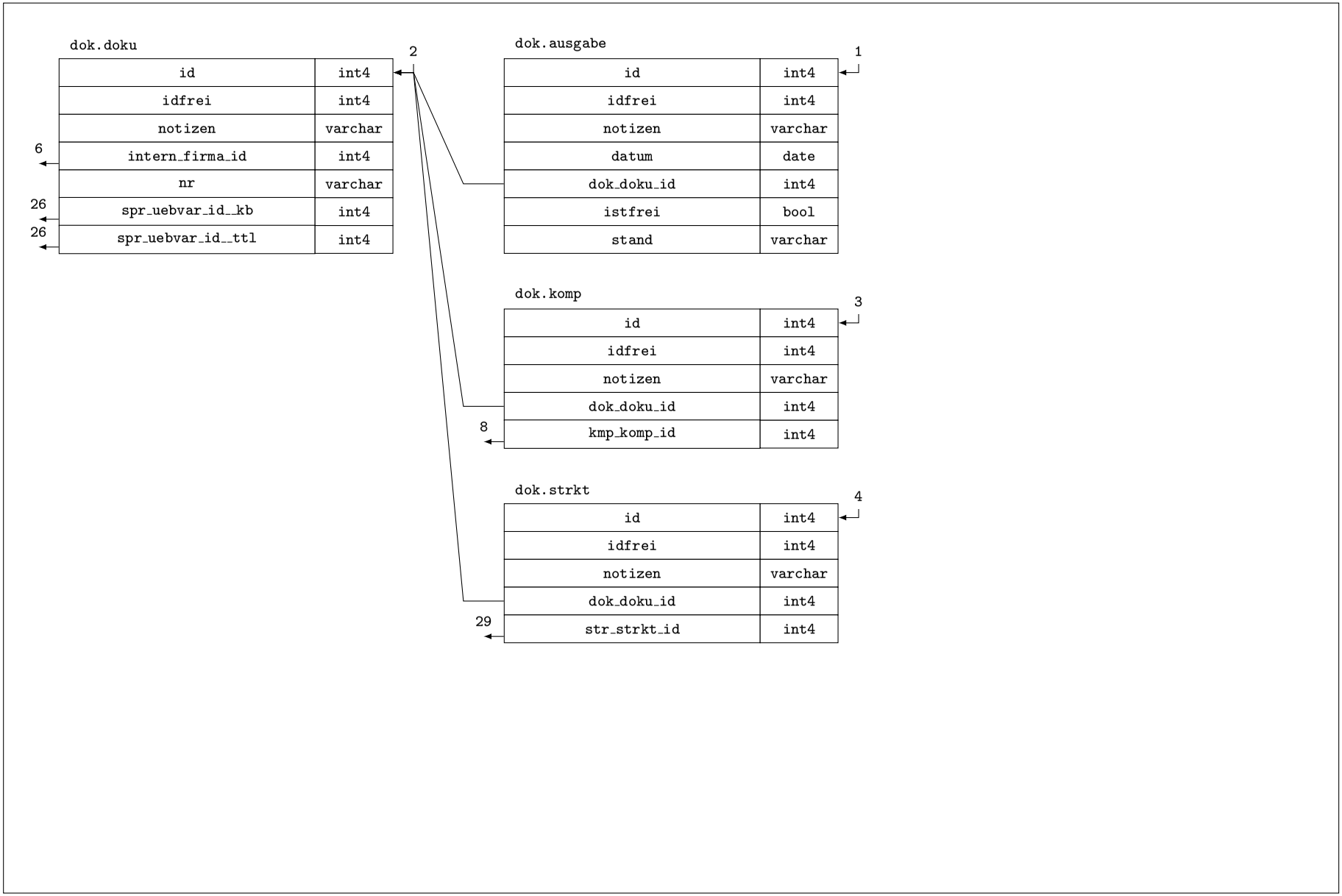
Die Namen der Sichten werden wie folgt festgelegt:

- `zeige__...` — Listen zurückgeben

3 dok

Im Schema „Dokumente“ werden die in der Datenbank verwendeten Dokumentenreferenzen gehalten.

3.1 Tabellenhierarchie der Ebenen 1 — 3



3.2 Schlüssel und Bedingungen

Primärschlüssel

pk__dok_ausgabe_id für Spalte dok.ausgabe.id

pk__dok_doku_id für Spalte dok.doku.id

pk__dok_komp_id für Spalte dok.komp.id

pk__dok_strkt_id für Spalte dok.strkt.id

Fremdschlüssel

fk__dok_ausgabe__dok_doku_id aus Spalte dok.ausgabe.dok_doku_id

fk__dok_doku__intern_firma_id aus Spalte dok.doku.intern_firma_id

fk__dok_doku__spr_uebvar_id__kb aus Spalte dok.doku.spr_uebvar_id__kb

fk__dok_doku__spr_uebvar_id__ttl aus Spalte dok.doku.spr_uebvar_id__ttl

fk__dok_komp__dok_doku_id aus Spalte dok.komp.dok_doku_id

fk__dok_komp__kmp_komp_id aus Spalte dok.komp.kmp_komp_id

fk__dok_strkt__dok_doku_id aus Spalte dok.strkt.dok_doku_id

fk__dok_strkt__str_strkt_id aus Spalte dok.strkt.str_strkt_id

Unique Constraints

uc__dok_ausgabe in Tabelle dok.ausgabe für die Spalte(n) dok.ausgabe.idfrei, dok.ausgabe.dok_d
dok.ausgabe.stand

uc__dok_doku in Tabelle dok.doku für die Spalte(n) dok.doku.idfrei, dok.doku.intern_firma_id,
dok.doku.nr

uc__dok_komp in Tabelle dok.komp für die Spalte(n) dok.komp.idfrei, dok.komp.dok_doku_id,
dok.komp.kmp_komp_id

uc__dok_strkt in Tabelle dok.strkt für die Spalte(n) dok.strkt.idfrei, dok.strkt.dok_doku_id,
dok.strkt.str_strkt_id

3.3 dok.nimm__doku

Funktionsname

```
dok.nimm__doku( IN liste_werte varchar[], OUT dokuid varchar )
```

Funktionsparameter/Rückgabewert(e)

liste_werte Liste der Werte, welche in dok.doku eingelesen werden sollen (Hinweise siehe Funktionsbeschreibung — Werteliste einlesen). Sie werden in der folgenden Reihenfolge in die Spalten

1. nr
2. intern_firma_id
3. spr_uebvar_id__ttl
4. spr_uebvar_id__kb
5. notizen

eingetragen. Es müssen mindestens die ersten zwei Werte übergeben werden.

dokuid Es wird die Datenbank-Identifikationsnummer der eingelesenen Dokumentenreferenz (dok.doku.id) zurückgegeben.

Funktionsbeschreibung
Mittels dieser Funktion werden die im System zu referenzierenden Dokumente in dok.doku eingelesen.

Werteliste einlesen:
Dem Funktionsparameter liste_werte ist eine Werteliste (ARRAY) zu übergeben (die maximale Anzahl und die Reihenfolge der Werte ist fest vorgegeben — siehe oben). Soll nur eine Teilmenge der Werte übergeben werden, dann sind alle Werte von links nach rechts zu spezifizieren (nichtbenötigte Werte am Ende der Liste dürfen weggelassen werden).

Beispiele
```
-- Fehlermeldung. Der Parameter liste_werte darf nicht NULL sein.
SELECT * FROM dok.nimm__doku( NULL );

-- Fehlermeldung. Es sind mindestens zwei Werte zu übergeben.
SELECT * FROM dok.nimm__doku( ARRAY['08-15'] );

-- Fehlermeldung. Die Werte müssen größer 0 bzw. dürfen nicht leer sein.
SELECT * FROM dok.nimm__doku( ARRAY['', '1'] );
SELECT * FROM dok.nimm__doku( ARRAY['08-15', '0'] );

-- Dokumentenreferenz einlesen
SELECT dok_doku_id, doknr, intern_firma_id__eigentuemer, spr_uebvar_id__doktitel,
spr_uebvar_id__kurzbeschr, dok_doku_notizen FROM dok.zeige__doku;
SELECT * FROM dok.nimm__doku( ARRAY['08-15', '1'] );
SELECT * FROM dok.nimm__doku( ARRAY['08-15', '1', spr.nimm__wgruppe( 'Dokumententitel',
'1', '0' )] );
SELECT * FROM dok.nimm__doku( ARRAY['08-15', '1', spr.nimm__wgruppe( 'Dokumententitel',
'1', '0' ), spr.nimm__wgruppe( 'Dokumentenkurzbeschreibung', '1', '0' )] );
SELECT * FROM dok.nimm__doku( ARRAY['08-15', '1', spr.nimm__wgruppe( 'Dokumententitel',
'1', '0' ), spr.nimm__wgruppe( 'Dokumentenkurzbeschreibung', '1', '0' ), 'Zusatzinformatio
);
SELECT dok_doku_id, doknr, intern_firma_id__eigentuemer, spr_uebvar_id__doktitel,
spr_uebvar_id__kurzbeschr, dok_doku_notizen FROM dok.zeige__doku;
```

3.4 dok.nimm__tabwerte

Funktionsname
`dok.nimm__tabwerte( IN tabname varchar, IN tabwerte varchar[], OUT tabid varchar )`

Funktionsparameter/Rückgabewert(e)

`tabname` Name der Tabelle (ohne Schemanamen) des Schemas `dok`, welche die Werte übernehmen soll (Details siehe Funktionsbeschreibung — Werteliste einlesen).

`tabwerte` Liste der Werte, welche in die aktuelle Tabelle eingelesen werden sollen (Details siehe Funktionsbeschreibung — Werteliste einlesen).

`tabid` Es wird die Identifikationsnummer des aktuell eingelesenen Wertes `dok.*.id` zurückgegeben.

Funktionsbeschreibung
Mittels dieser Funktion werden in die Tabellen des Schemas `dok` die Werte eingelesen.

Werteliste einlesen:
Dem Funktionsparameter `tabwerte` ist eine Werteliste (`ARRAY`) zu übergeben (die maximale Anzahl und die Reihenfolge der Werte ist fest vorgegeben — siehe unten). Soll nur eine Teilmenge der Werte übergeben werden, dann sind alle Werte von links nach rechts zu spezifizieren (nichtbenötigte Werte am Ende der Liste dürfen weggelassen werden).

ausgabe, komp, strkt

1. `dok_doku_id`
2. `stand, kmp_komp_id, str_strkt_id` Für die Spalten `*_id` muß der Wert größer `0` sein.
3. `istfrei, notizen, notizen`
4. `datum, -, -`
5. `notizen, -, -`

Es müssen mindestens die ersten beiden Werte übergeben werden. Der erste Wert muß größer `0` sein.

Beispiele
```
-- Fehlermeldung. Ungültige Tabelle !
SELECT * FROM dok.nimm__tabwerte( NULL, NULL );

-- Fehlermeldung. Bei allen Tabellen müssen mindestens zwei Elemente übergeben
werden
SELECT * FROM dok.nimm__tabwerte( 'ausgabe', ARRAY['1'] );
```

```
-- Fehlermeldung. Die übergebenen Parameter müssen größer 0 sein.
SELECT * FROM dok.nimm__tabwerte( 'ausgabe', ARRAY['0', '001'] );
SELECT * FROM dok.nimm__tabwerte( 'komp', ARRAY['1', '0'] );
SELECT * FROM dok.nimm__tabwerte( 'strkt', ARRAY['1', '0'] );

-- zum Testen für alle Schemas aktivieren
SELECT * FROM dok.nimm__doku( ARRAY['08-15', '1'] );

-- Revisionsstand zum Dokument einlesen
SELECT * FROM dok.ausgabe;
SELECT * FROM dok.nimm__tabwerte( 'ausgabe', ARRAY['1', '01', 'false', '2007-05-08',
'Zustazinformationen'] );
SELECT * FROM dok.ausgabe;

-- Komponente einlesen
SELECT * FROM dok.komp;
SELECT * FROM dok.nimm__tabwerte( 'komp', ARRAY['1', kmp.nimm__komp( ARRAY['1',
'47/11'] ), 'Zustazinformationen'] );
SELECT * FROM dok.komp;

-- Strukturpunkt einlesen
SELECT * FROM dok.strkt;
SELECT * FROM dok.nimm__tabwerte( 'strkt', ARRAY['1', str.nimm__strkt( ARRAY['1',
'-BAAA', spr.nimm__wgruppe( 'Strukturpunkt', '1', '0' )] ), 'Zusatzinformationen']
);
SELECT * FROM dok.strkt;
```

3.5 dok.zeige__doku

Beschreibung der Spalten

dok_doku_id siehe dok.doku.id

doknr siehe dok.doku.nr

intern_firma_id__eigentuemer siehe dok.doku.intern_firma_id

spr_uebvar_id__doktitel siehe dok.doku.spr_uebvar_id__ttl

spr_uebvar_id__kurzbeschr siehe dok.doku.spr_uebvar_id__kb

dok_doku_notizen siehe dok.doku.notizen

dok_ausgabe_id siehe dok.ausgabe.id

dokausgabe siehe dok.ausgabe.stand

`istfreigegeben` siehe `dok.ausgabe.istfrei`

`dok_ausgabe_datum` siehe `dok.ausgabe.datum`

`dok_ausgabe_notizen` siehe `dok.ausgabe.notizen`

`dok_komp_id` siehe `dok.komp.id`

`kmp_komp_id` siehe `dok.komp.kmp_komp_id`

`dok_komp_notizen` siehe `dok.komp.notizen`

`dok_strkt_id` siehe `dok.strkt.id`

`str_strkt_id` siehe `dok.strkt.str_strkt_id`

`dok_strkt_notizen` siehe `dok.strkt.notizen`

Beschreibung der Sicht
Es werden sämtliche Informationen zu den Dokumenten zurückgegeben.

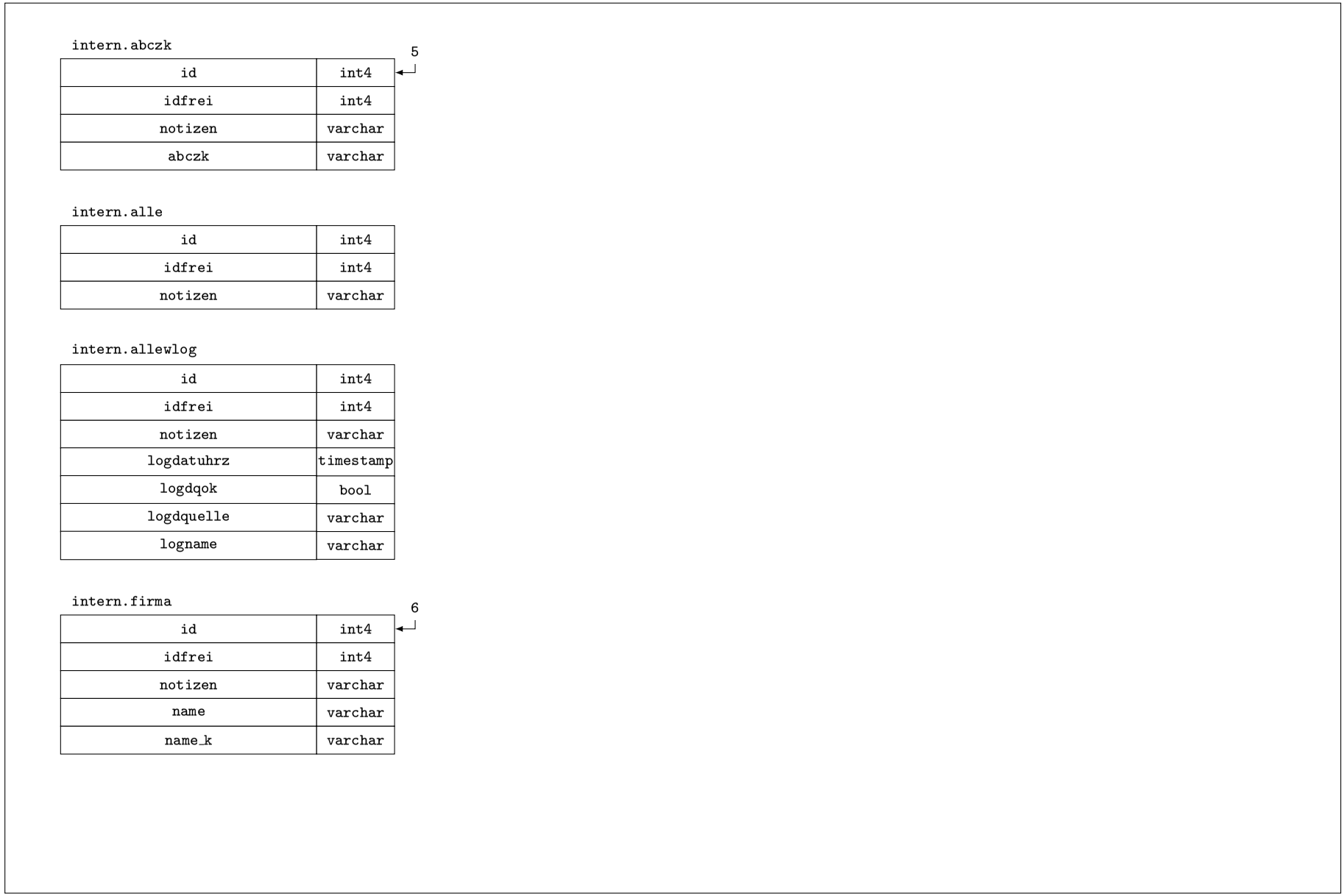
4 intern

Das Schema `Intern` dient der Bereitstellung von Tabellen, Datenstrukturen, usw. welche in allen Schemas verwendet werden können. Weiterhin wird die Basis der Logbuch-Funktionalität einschließlich der Logbuch-Funktion der Tabellen

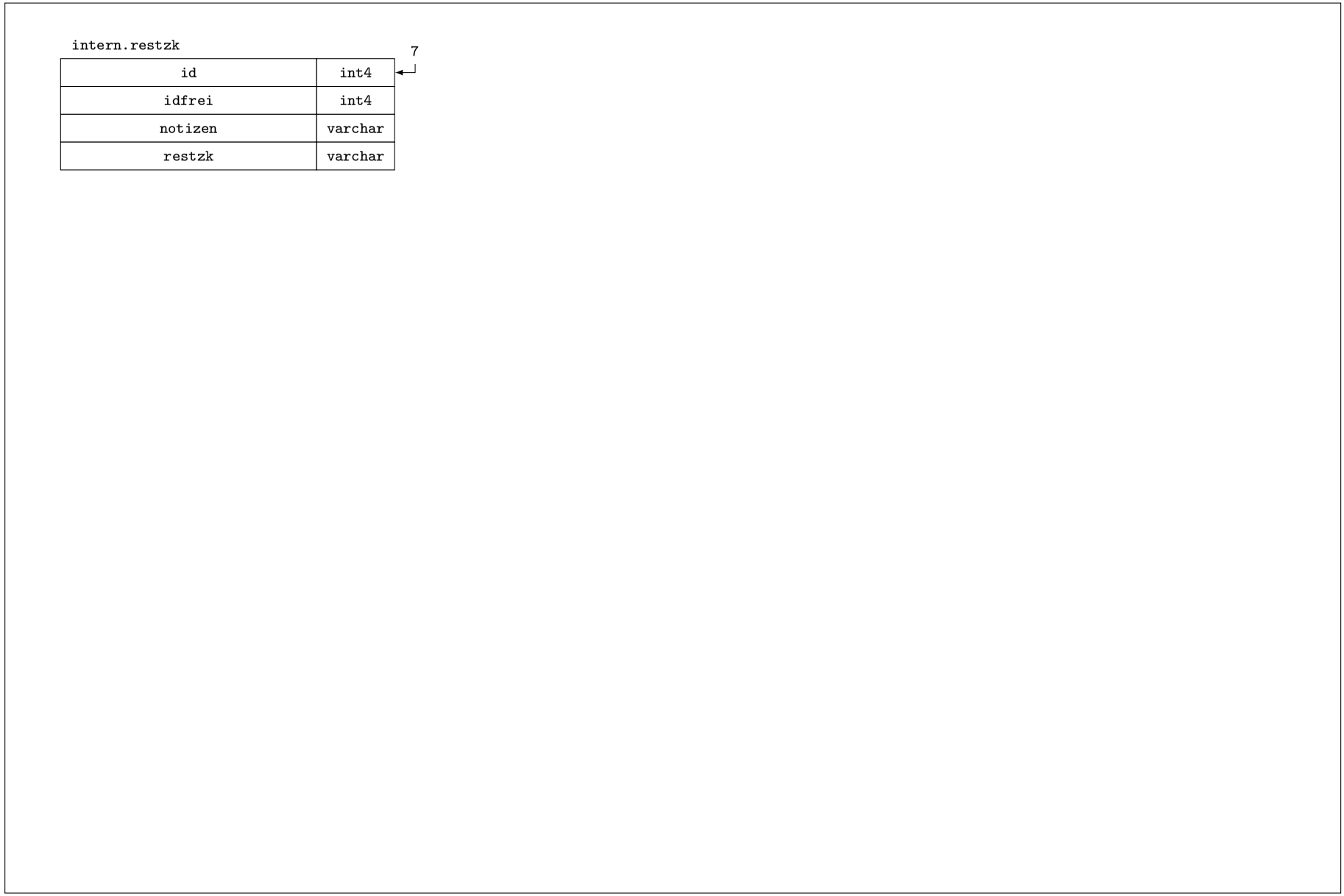
- `abczk`,
- `restzk`

bereitgestellt.

4.1 Tabellenhierarchie der Ebenen 1 — 3



4.2 Tabellenhierarchie der Ebenen 1 — 3



4.3 Schlüssel und Bedingungen

Primärschlüssel

pk__intern_abczk_id für Spalte intern.abczk.id

pk__intern_firma_id für Spalte intern.firma.id

pk__intern_restzk_id für Spalte intern.restzk.id

Unique Constraints

uc__intern_abczk in Tabelle intern.abczk für die Spalte(n) intern.abczk.idfrei, intern.abczk.a

uc__intern_firma in Tabelle intern.firma für die Spalte(n) intern.firma.idfrei, intern.firma.n
intern.firma.name_k

uc__intern_restzk in Tabelle intern.restzk für die Spalte(n) intern.restzk.idfrei,
intern.restzk.restzk

Check Constraints

cc__intern_alle in Tabelle intern.alle die Bedingung ((id = 0) AND (idfrei = 0))

cc__intern_allewlog in Tabelle intern.allewlog die Bedingung ((((((id = 0) AND (idfrei
= 0)) AND (logdatuhrz IS NOT NULL)) AND (logname IS NOT NULL)) AND (logdquelle
IS NULL)) AND (logdqok IS NULL))

4.4 intern.aggregat__zk

Funktionsname
intern.aggregat__zk()

Funktionsparameter/Rückgabewert(e)

-

Funktionsbeschreibung
Diese Aggregatfunktion verbindet sämtliche Teilzeichenketten zu einer Zeichenkette.

4.5 intern.gib__ere_abczk

Funktionsname
intern.gib__ere_abczk(OUT ere_abczk varchar)

Funktionsparameter/Rückgabewert(e)

ere_abczk Ist der erweiterte regulaere Ausdruck zur Beschreibung einer AbcZK.

Funktionsbeschreibung
Ziel der Funktion ist das Definieren und Zurückgeben eines erweiterten regulären Ausdruckes (ERE), welcher das Textmuster eine Zeichenkette beschreibt, die ausschließlich aus Buchstaben besteht - (AbcZK)- (siehe `intern.gib__ERE_Buchstabe(...)`).

Beispiele
```
-- ERE zurückgeben
SELECT * FROM intern.gib__ERE_ABCZK();

-- hallo zurückgeben
SELECT substring( 'hallo' from intern.gib__ERE_ABCZK() );

-- hal zurückgeben
SELECT substring( 'hal2lo' from intern.gib__ERE_ABCZK() );
```

4.6 intern.gib__ere_buchstabe

Funktionsname
`intern.gib__ere_buchstabe( IN istanfang bool, OUT ere_buchstabe varchar )`

Funktionsparameter/Rückgabewert(e)

`istanfang` Der erweiterte reguläre Ausdruck eines Anfangsbuchstaben soll zurückgegeben werden

`ere_buchstabe` Soll der erweiterte reguläre Ausdruck zur Beschreibung eines Anfangsbuchstaben zurückgegeben werden, ist `istAnfang = true` zu setzen. Soll der erweiterte reguläre Ausdruck zur Beschreibung eines Folgebuchstaben zurückgegeben werden, ist `istAnfang = false` zu setzen.

Funktionsbeschreibung
Ziel der Funktion ist das Definieren und Zurueckgeben eines erweiterten regulären Ausdruckes (ERE), welcher das Textmuster eines Anfangsbuchstaben oder Folgebuchstabens beschreibt. Die folgenden Buchstaben werden vom ERE erfaßt:

- A-Z
- a-z
- äöü
- ÄÖÜ
- ß (nur Folgebuchstabe)

Beispiele

```
-- ERE für Anfangsbuchstaben zurückgeben
SELECT * FROM intern.gib__ERE_Buchstabe( true );

-- ERE für die Buchstaben im Wort zurückgeben
SELECT * FROM intern.gib__ERE_Buchstabe( false );

-- h zurückgeben
SELECT substring( 'h' from intern.gib__ERE_Buchstabe( true ) );

-- Leerzeichenkette zurückgeben
SELECT substring( 'ß' from intern.gib__ERE_Buchstabe( true ) );

-- ß zurückgeben
SELECT substring( 'ß' from intern.gib__ERE_Buchstabe( false ) );
```

4.7 intern.gib__ere_restzk

Funktionsname

`intern.gib__ere_restzk( OUT ere_restzk varchar )`

Funktionsparameter/Rückgabewert(e)

`ere_restzk` Ist der erweiterte reguläre Ausdruck zur Beschreibung der Zeichenketten, die keine Buchstaben enthalten (in `intern.restzk` gültig sind).

Funktionsbeschreibung

Ziel der Funktion ist das Definieren und Zurückgeben eines erweiterten regulären Ausdruckes (ERE), welcher das Textmuster der Zeichenkette beschreibt, die über die Tastatur eingebbar ist, aber nicht aus Buchstaben besteht.

Beispiele

```
-- ERE zurückgeben
SELECT * FROM intern.gib__ERE_RestZK();
```

4.8 intern.gib__liste_abczkrestzk

Funktionsname

`intern.gib__liste_abczkrestzk( IN wortgruppe varchar, OUT zk varchar, OUT istabczk bool )`

Funktionsparameter/Rückgabewert(e)

`wortgruppe` Wortgruppe, welche in `abczk` bzw. `restzk` zerlegt werden soll

`zk` Zeichenkette aus der `wortgruppe`, welche eine `abczk` oder `restzk` ist

`istabczk` Wenn `istabczk = true` dann ist die in `zk` vorliegende Zeichenkette eine `abczk` sonst ist sie eine `restzk`

Funktionsbeschreibung
Ziel der Funktion ist das Zerlegen der Wortgruppe `wortgruppe` in `abczk` bzw. `restzk` mittels Erweiterter Regulaerer Ausdruecke (ERE).

Die Vorschrift zum Erkennen einer

- `abczk` kann `intern.gib__ERE_ABCZK()` bzw.
- `restzk` kann `intern.gib__ERE_RestZK()`

entnommen werden.

Beispiele
```
-- Fehlermeldung. Die Wortgruppe ist anzugeben.
SELECT * FROM intern.gib__Liste_abczkrestzk( NULL );
SELECT * FROM intern.gib__Liste_abczkrestzk( '' );

-- Es werden die Bestandteile der Wortgruppe zurückgegeben
SELECT '>' || zk || '<' AS zk, istabczk FROM intern.gib__Liste_abczkrestzk( 'Das
ist ein Array - This is an Array - 1984' );
```

4.9 intern.gib__liste_tabellenpositionen

Funktionsname
`intern.gib__liste_tabellenpositionen( IN schname varchar, OUT tabname varchar, OUT ebene varchar )`

Funktionsparameter/Rückgabewert(e)

`schname` Name des Schemas dessen Tabellenpositionen (Ebenen innerhalb der Tabellenhierarchie) ermittelt werden.

`tabname` Tabellenname des Schemas

`ebene` Position (Ebene) der Tabelle im Schema

Funktionsbeschreibung
Ziel der Funktion ist das Ermitteln der Positionen (Ebenen) sämtlicher Tabellen im Baum der Tabellenverbindungen (Fremdschlüssel) des Schemas `schName`.

Beispiele
```
-- Fehlermeldung. Der Parameter schName darf nicht NULL oder Leer sein.
```

```
SELECT * FROM intern.gib__Liste_Tabellenpositionen( '' );

-- Tabellenpositionen ermitteln
SELECT * FROM intern.gib__Liste_Tabellenpositionen( 'dok' );
SELECT * FROM intern.gib__Liste_Tabellenpositionen( 'intern' );
SELECT * FROM intern.gib__Liste_Tabellenpositionen( 'kmp' );
SELECT * FROM intern.gib__Liste_Tabellenpositionen( 'prj' );
SELECT * FROM intern.gib__Liste_Tabellenpositionen( 'prt' );
SELECT * FROM intern.gib__Liste_Tabellenpositionen( 'ps' );
SELECT * FROM intern.gib__Liste_Tabellenpositionen( 'spr' );
SELECT * FROM intern.gib__Liste_Tabellenpositionen( 'str' );
SELECT * FROM intern.gib__Liste_Tabellenpositionen( 'tl' );
```

4.10 intern.gib__naechsteid

Funktionsname
`intern.gib__naechsteid( IN tabname varchar, OUT tabnameid varchar )`

Funktionsparameter/Rückgabewert(e)

`tabname` Name der Tabelle (ggf. einschliesslich Schemaname), aus der die nächste verfügbare Identifikationsnummer `id` ermittelt werden soll. (darf nicht leer sein).

`tabnameid` Nächste freie `id`

Funktionsbeschreibung
Ziel der Funktion ist das Ermitteln der nächsten verfügbaren `id` der übergebenen Tabelle `tabName`. Es wird der kleinste mögliche Wert gesucht.

Beispiele
```
-- Fehlermeldung. Der Parameter tabName darf nicht NULL sein.
SELECT intern.gib__naechsteid( NULL );

-- Eine wiederverwendbare id wurde nicht gefunden
SELECT max(id) FROM intern.abczk;
SELECT intern.gib__naechsteid( 'intern.abczk' );

-- Eine wiederverwendbare id wurde gefunden
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['abczk'], ARRAY['KeinRichtigesWort']
);
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['abczk'], ARRAY['KeinRichtigesWortZwe
);
SELECT * FROM intern.abczk ORDER BY id DESC LIMIT 5;
SELECT intern.loesche__zeile( 'intern.abczk', 'abczk = ''KeinRichtigesWort''' );
SELECT * FROM intern.abczk ORDER BY id DESC LIMIT 5;
```

```
SELECT intern.gib__naechsteid( 'intern.abczk' );
```

4.11 intern.gib__sqlselect

Funktionsname
`intern.gib__sqlselect( IN sel varchar, IN sel_from varchar[], IN sel_where varchar, OUT sqlselect varchar )`

Funktionsparameter/Rückgabewert(e)

`sel` Es ist die Liste der Parameter anzugeben, welche das Ergebnis der `SELECT` - Abfrage sind. Diese Variable darf nicht leer sein.

`sel_from` Es ist die Liste (`ARRAY`) der Tabellen anzugeben, welche mittels `SELECT` - Abfrage ausgewertet werden sollen. Diese Variable darf nicht leer sein.

`sel_where` Es ist eine `WHERE` - Bedingung anzugeben (ggf. einschließlich Semikolon). Diese Variable darf leer sein.

`sqlselect` Es wird die `SELECT` - Anweisung zurueckgegeben.

Funktionsbeschreibung
Ziel der Funktion ist das Erzeugen einer `SELECT` - Anweisung, welche innerhalb einer `PL/pgSQL` - Funktion verwendet werden kann. Sie berücksichtigt das Konzept zur Wiederverwendung gelöschter `id`. Das abschliessende Semikolon ist explizit anzugeben.

Beispiele
```
-- Fehlermeldung. Der Parameter sel darf nicht leer oder NULL sein.
SELECT intern.gib__sqlSELECT( '', '', '' );

-- Fehlermeldung. Der Parameter sel_from darf nicht leer oder NULL sein.
SELECT intern.gib__sqlSELECT( 'hsk', NULL, '' );

-- Fehlermeldung. Kein Element der Liste sel_from darf leer sein
SELECT intern.gib__sqlSELECT( 'hsk', ARRAY[''], '' );

-- SQL-Abfrage erzeugen
SELECT intern.gib__sqlSELECT( 'id, idfrei, notizen', 'intern.alle', '' );
SELECT intern.gib__sqlSELECT( 'id, idfrei, notizen', 'tt.muell, tt.test', ' AND
tt.muell.id = tt.test.tt_muell_id' );
```

4.12 intern.gib__stsname

Funktionsname
`intern.gib__stsname( IN befehl varchar, IN vollstname varchar, OUT name varchar`

)

Funktionsparameter/Rückgabewert(e)

`befehl` In Form eines fest vorgegebenen Befehls ist anzugeben, welcher Teil des vollständigen Spaltennamens zurückzugeben ist:

sch Schemaname

tab Tabellenname

sp Spaltenname

schtab Schemaname und Tabellenname

tabsp Tabellenname und Spaltenname

Die Gültigkeit der übergebenen Befehle wird nicht überprüft. Es darf nicht `NULL` bzw. keine Leerzeichenkette angegeben werden.

`vollstname` Es ist der vollständige Spaltenname anzugeben. Es wird geprüft, ob die Syntax eingehalten wurde.

`name` Es wird der Name zurückgebene, welcher in `befehl` spezifiziert wurde.

Funktionsbeschreibung
Ziel der Funktion ist das Extrahieren des Schema,- Tabellen- oder Spaltennamens aus dem vollständigen Spaltennamen. Der vollständige Spaltenname besteht aus dem Schemanamen, dem Tabellennamen und dem Spaltennamen welche mittels Punkt getrennt sind und aus Buchstaben `a-z` bzw. `A-Z`, Ziffern `0-9` bzw. Unterstrich bestehen.
Im Fehlerfall wird eine Leerzeichenkette zurückgegeben.

Beispiele

```
-- Fehlermeldung. Es ist ein vollständiger Spaltenname anzugeben
SELECT * FROM intern.gib__stsname( 'sp', NULL );
SELECT * FROM intern.gib__stsname( 'sp', '' );

-- Fehlermeldung. Es ist Befehl anzugeben
SELECT * FROM intern.gib__stsname( NULL, 'intern.alle.notizen' );
SELECT * FROM intern.gib__stsname( '', 'intern.alle.notizen' );

-- Fehlermeldung. Der vollständige Spaltenname ist in der Syntax schema.tabelle.spalte
anzugeben
SELECT * FROM intern.gib__stsname( 'sch', 'i.a' );
SELECT * FROM intern.gib__stsname( 'sch', '.a.n' );
SELECT * FROM intern.gib__stsname( 'sch', 'i..n' );
SELECT * FROM intern.gib__stsname( 'sch', 'i.a.' );
SELECT * FROM intern.gib__stsname( 'sch', '..' );
SELECT * FROM intern.gib__stsname( 'sch', 'iß.a.n' );
SELECT * FROM intern.gib__stsname( 'sch', 'i.a2.n' );
SELECT * FROM intern.gib__stsname( 'sch', 'i.a.n,' );
```

```
SELECT * FROM intern.gib__stsname( 'sch', 'i.a.n.falsch' );

-- Fehlermeldung. Der übergebene befehl ist falsch
SELECT * FROM intern.gib__stsname( 'unbekannterBefehl', 'intern.alle.notizen' );

-- Schema, Tabelle, Spalte, ... zurückgeben
SELECT * FROM intern.gib__stsname( 'sch', 'intern.alle.notizen' );
SELECT * FROM intern.gib__stsname( 'tab', 'intern.alle.notizen' );
SELECT * FROM intern.gib__stsname( 'sp', 'intern.alle.notizen' );
SELECT * FROM intern.gib__stsname( 'schtab', 'intern.alle.notizen' );
SELECT * FROM intern.gib__stsname( 'tabsp', 'intern.alle.notizen' );
SELECT * FROM intern.gib__stsname( 'tabsp', 'int2ern.al_le.notizen85' );
```

4.13 intern.gib__zu

Funktionsname
`intern.gib__zu( OUT zu bpchar )`

Funktionsparameter/Rückgabewert(e)

`zu` Das Ersatzzeichen für den Zeilenumbruch

Funktionsbeschreibung
Ziel der Funktion ist das Zurückgeben des im System zugelassenen Ersatzzeichens für den Zeilenumbruch.
Alle Texte, welche ein Zeilenumbruchzeichen enthalten, sollen mit dem hier zurückgegebenen Zeichen erstellt (gespeichert werden). Das Bearbeiten der Datenbankdumps oder der Texte, welche aus der Datenbank ausgelesen werden, wird vereinfacht.

Beispiele
```
-- Es wird das Ersatzzeichen für den Zeilenumbruch zurückgegeben
SELECT * FROM intern.gib__ZU();
```

4.14 intern.ist__abczk

Funktionsname
`intern.ist__abczk( IN abczk varchar, OUT istabczk bool )`

Funktionsparameter/Rückgabewert(e)

`abczk` Es ist die zu testende Zeichenkette anzugeben.

`istabczk` Wenn eine AbcZK vorliegt wird `true` zurückgegeben, sonst `false`.

Funktionsbeschreibung
Ziel der Funktion ist das Prüfen, ob die übergebene Zeichenkette `abczk` als Zeichenkette interpretiert werden kann, welche ausschließlich aus Buchstaben besteht (AbcZK). Die gültigen Buchstaben können `intern.gib__ERE_ABCZK()` entnommen werden.

Beispiele
```
-- Es wird false zurückgegeben
SELECT * FROM intern.ist__abczk( '' );
SELECT * FROM intern.ist__abczk( 'ein-/auslesen' );
SELECT * FROM intern.ist__abczk( 'ß' );
SELECT * FROM intern.ist__abczk( 'ßbeginnt' );

-- Es wird true zurückgegeben
SELECT * FROM intern.ist__abczk( 'I' );
SELECT * FROM intern.ist__abczk( 'daß' );
```

4.15 intern.loesche__zeile

Funktionsname
`intern.loesche__zeile( IN tabname varchar, IN sql_where varchar )`

Funktionsparameter/Rückgabewert(e)

`tabname` Name einschliesslich Schemaname der Tabelle, aus der die Zeile gelöscht werden soll (darf nicht leer sein)

`sql_where` SQL-Bedingung welche die zu löschende(n) Zeile(n) beschreibt (darf leer sein)

Funktionsbeschreibung
Ziel der Funktion ist das Löschen von Zeilen (außer der Zeile mit `id = 0`) unter Berücksichtigung der Bedingungen, welche mittels `sql_where` spezifiziert sind. Wenn `sql_where` leer ist, werden alle Zeilen gelöscht.
Es ist besonders zu beachten, daß `tabName` den Schemanamen enthalten muss !

Beispiele
```
-- Fehlermeldung. Der Tabellenname ist anzugeben
SELECT intern.loesche__zeile( NULL, '' );

-- Fehlermeldung. Die übergebene Tabelle ist nicht vorhanden
SELECT intern.loesche__zeile( 'intern.abc', '' );

-- Fehlermeldung. Die Zeilen können nicht gelöscht werden, da weitere Zeilen abhängen.
SELECT intern.loesche__zeile( 'intern.abczk', '' );

-- Fehlermeldung. Die übergebe WHERE - Bedingung ist ungültig
```

```
SELECT intern.loesche__zeile( 'intern.abczk', 'intern.abczk.id = 99999999' );

-- Tabelleninhalt anzeigen
SELECT intern.nimm__zeile( 'tl.datei', ARRAY['dname'], ARRAY['Dateiname-1']);
SELECT intern.nimm__zeile( 'tl.datei', ARRAY['dname'], ARRAY['Dateiname-2']);
SELECT * FROM tl.datei WHERE id < 5;

-- Gesamten Inhalt der Tabelle löschen
SELECT intern.loesche__zeile( 'tl.datei', NULL );

-- Zeile mit der id = 1 löschen
SELECT intern.loesche__zeile( 'tl.datei', 'tl.datei.id = 1' );

-- Tabelleninhalt anzeigen
SELECT * FROM tl.datei WHERE id < 5;
```

4.16 intern.nimm__tlkommentar

Funktionsname
`intern.nimm__tlkommentar( IN sqldbobj varchar, IN kommentar varchar )`

Funktionsparameter/Rückgabewert(e)

`sqldbobj` Es ist das Datenbankobjekt zu benennen (SQL) welchem der Kommentar hinzugefügt werden soll.

`kommentar` Der Kommentar (einschl. TL-Befehlen). '\' sind zu maskieren !

Funktionsbeschreibung
Ziel der Funktion ist das Hinzufügen eines Kommentares zu einem Datenbankobjekt. Im Kommentar können TL-Befehle verwendet werden, da die Dokumentation zur Datenbank mittels TEX/LATEX formatiert wird.
Zeilenumbrüche werden mittels `intern.gib__ZU()` ersetzt.

Beispiele
```
-- Fehlermeldung. Der Parameter sqlDBObj darf nicht leer oder NULL sein.
SELECT intern.nimm__TLKommentar( '', '' );
SELECT intern.nimm__TLKommentar( NULL, '' );

-- Kommantar gelöscht
SELECT intern.nimm__TLKommentar( 'DATABASE ' || current_database(), '' );

-- Kommantar hinzugefügt
SELECT intern.nimm__TLKommentar( 'DATABASE ' || current_database(), 'Neuer Kommentar'
);
```

```
-- prüfen, ob Kommantar tatsächlich hinzugefügt wurde
SELECT shobj_description( ( SELECT oid FROM pg_database WHERE pg_database.datname
= current_database() LIMIT 1 ), 'pg_database');
```

4.17 intern.nimm__tlkommentarfunktion

Funktionsname
`intern.nimm__tlkommentarfunktion( IN fkt varchar, IN fktparam varchar[], IN beschr varchar, IN beispiel varchar[] )`

Funktionsparameter/Rückgabewert(e)

`fkt` Es ist der Name der Funktion anzugeben, der der Kommentar zugeordnet werden soll. Er darf nicht leer oder NULL sein.

`fktparam` In Form eines ARRAY sind in der Reihenfolge ihres Auftretens sämtliche Beschreibungen der Funktionsparameter anzugeben. Es wird nicht zwischen Eingabeparametern `IN` bzw. Rückgabeparametern `OUT / INOUT` unterschieden. Sollte die Funktion über keine Parameter verfügen, ist `NULL` anzugeben.

`beschr` Es ist die Beschreibung der Funktion einzugeben. Sie darf nicht leer oder `NULL` sein.

`beispiel` Es können mehrere Beispiele zur Demonstration der Nutzung der Funktion angegeben werden. Sie dürfen `NULL` sein.

Funktionsbeschreibung
Ziel der Funktion ist das Hinzufügen einer Beschreibung (eines Kommentars) zu einer Funktion. Hinweise zur Gültigkeit können `intern.nimm__TLKommentar(...)` entnommen werden.

Beispiele
```
-- Fehlermeldung. Die Parameter fkt und beschr dürfen nicht NULL sein.
SELECT intern.nimm__TLKommentarFunktion( NULL, NULL, '', NULL );
SELECT intern.nimm__TLKommentarFunktion( 'intern.nimm__TLKommentarFunktion', NULL,
'', NULL );

-- Kommentar einlesen
SELECT intern.nimm__TLKommentarFunktion( 'intern.nimm__TLKommentarFunktion( fkt
varchar, fktparam varchar[], beschr varchar, beispiel varchar[] )', NULL, 'Testbeschre
NULL );
\df+ intern.nimm__tlkommentarfunktion
```

4.18 intern.nimm__tlkommentarsicht

Funktionsname
`intern.nimm__tlkommentarsicht( IN sicht varchar, IN spnamen varchar[], IN spbeschr varchar[], IN beschr varchar )`

Funktionsparameter/Rückgabewert(e)

`sicht` Es ist der vollständige Name der Sicht anzugeben. Er darf nicht leer oder NULL sein.

`spnamen` In Form eines `ARRAY` sind in der Reihenfolge ihres Auftretens sämtliche Namen der Spalten anzugeben, welche von der Sicht bereitgestellt werden. Es muß mindestens eine Spalte beschrieben werden.

`spbeschr` In Form eines `ARRAY` sind in der Reihenfolge ihres Auftretens sämtliche Beschreibungen der Spalten anzugeben, welche von der Sicht bereitgestellt werden. Es muß mindestens eine Spalte beschrieben werden.

`beschr` Es ist die Sicht zu beschreiben. Es muß eine Beschreibung angegeben werden.

Funktionsbeschreibung
Ziel der Funktion ist das Hinzufügen einer Beschreibung (eines Kommentars) zu einer Sicht. Hinweise zur Gültigkeit können `intern.nimm__TLKommentar(...)` entnommen werden.

Beispiele
```
-- Fehlermeldung. Der Name der Sicht fehlt
SELECT intern.nimm__TLKommentarSicht( NULL, NULL, NULL, '' );

-- Fehlermeldung. Der Kommentar fehlt
SELECT intern.nimm__TLKommentarSicht( 'intern.zeige__fremdschluessel', ARRAY['spName'],
ARRAY['spBeschreibung'], '' );

-- Fehlermeldung. Es ist mindestens eine Spalte zu beschreiben
SELECT intern.nimm__TLKommentarSicht( 'intern.zeige__fremdschluessel', NULL, ARRAY['spBesch
'Kommentar' );
SELECT intern.nimm__TLKommentarSicht( 'intern.zeige__fremdschluessel', ARRAY['spName'],
NULL, 'Kommentar' );

-- Fehlermeldung. Die Anzahl der Spaltennamen muß mit der Anzahl der Spaltenbeschreibunge
übereinstimmen
SELECT intern.nimm__TLKommentarSicht( 'intern.zeige__fremdschluessel', ARRAY['spName1',
'spName2'], ARRAY['spBeschreibung'], 'Kommentar' );

-- Sicht kommentieren, Ergebnis überprüfen
SELECT intern.nimm__TLKommentarSicht( 'intern.zeige__fremdschluessel', ARRAY['SpName'],
ARRAY['SpBeschreibung'], 'Beschreibung' );
\dv+ intern.zeige__fremdschluessel
```

4.19 intern.nimm__tlkommentarsts

Funktionsname
`intern.nimm__tlkommentarsts( IN sts varchar, IN kommentar varchar )`

Funktionsparameter/Rückgabewert(e)

`sts` Es ist der vollständige Schemaname, Tabelnname oder Spaltenname anzugeben.

`kommentar` Es ist der Kommentar anzugeben

Funktionsbeschreibung
Ziel der Funktion ist das Hinzufügen eines Kommentares zu einem Schema, einer Tabelle oder einer Spalte. Es ist der vollständige Name anzugeben (Bsp.: `schema.tabelle.spalte`). Anhand der Anzahl der Datenbankobjekte wird entschieden, ob ein Schema, eine Tabelle oder eine Spalte vorliegt.
Hinweise zur Gültigkeit können `intern.nimm__TLKommentar(...)` entnommen werden.

Beispiele
```
-- Fehlermeldung. Der Parameter sts darf nicht NULL sein.
SELECT intern.nimm__TLKommentarSTS( '', '' );
SELECT intern.nimm__TLKommentarSTS( NULL, '' );

-- Kommantar gelöscht
SELECT intern.nimm__TLKommentarSTS( 'intern.alle.id', '' );

-- Kommantar hinzugefügt
SELECT intern.nimm__TLKommentarSTS( 'intern.alle.id', 'Neuer Kommentar' );
\d+ intern.alle
```

4.20 intern.nimm__zeile

Funktionsname
`intern.nimm__zeile( IN tabname varchar, IN liste_spname varchar[], IN liste_werte varchar[], OUT tabnameid varchar )`

Funktionsparameter/Rückgabewert(e)

`tabname` Name der Tabelle (ggf. einschliesslich Schemaname), in die die Werte `liste_werte[]` eingelesen werden sollen (darf nicht leer sein).

`liste_spname` Liste der Spalten, welche die Werte aufnehmen sollen. Es muß mindestens ein Spaltenname übergeben werden. Es dürfen keine Leerzeichenketten übergeben werden.

`liste_werte` Liste der Werte, welche eingefügt werden sollen. Es muß mindestens ein Wert übergeben werden. Es dürfen keine Leerzeichenketten übergeben werden. Weiterhin muß dieselbe Anzahl Werte übergeben werden, wie Spalten benannt sind.

`tabnameid id` der Zeile, welche die übergebenen Werte aufgenommen hat

Funktionsbeschreibung
Ziel der Funktion ist das Einlesen von Werten. Der Spaltenliste `liste_spName[]` darf jede in der Tabelle `tabName` vorhandene Spalte übergeben werden. Die Spalte `idfrei` darf nicht pbergeben werden, da sie ausschließlich vom System gesetzt wird.
Die übergebenen Werte `liste_werte[]` werden in der DB gesucht und die `id` der Zeile zurückgegeben, welche die Werte bereits enthält. Sollte keine Zeile gefunden werden, wird die nächste verfügbare `id` ermittelt und die Werte eingelesen.

Sollte `liste_spName[]` die Spalte `id` explizit benannt sein, so wird unabhängig vom Wert der Spalte `idfrei` die übergebene `id` gesucht, der gesamte Inhalt gelöscht und die neuen Werte eingetragen. Wurde `id` nicht gefunden, wird eine Zeile mit der uebergebenen `id` hinzugefügt.

Beispiele

```
-- Fehlermeldung. Der Parameter tabName darf nicht NULL oder leer sein.
SELECT intern.nimm__zeile( NULL, ARRAY[''], ARRAY[''] );

-- Fehlermeldung. Die Parameter liste_spName und liste_werte müssen mindestens
ein Element enthalten.
SELECT intern.nimm__zeile( 'intern.abczk', NULL, NULL );
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['abczk'], NULL );

-- Fehlermeldung. Die Parameter liste_spName und liste_werte dürfen keine Leerzeichenkette
enthalten.
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY[''], ARRAY[''] );
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['abczk'], ARRAY[''] );

-- Fehlermeldung. Der Parameter liste_wert muß dieselbe Anzahl Elemente enthalten,
wie der Parameter liste_spName.
SELECT intern.nimm__zeile( 'intern.abczk', 'notizen, abczk', 'zeichenkette für
notizen' );

-- Fehlermeldung. Die Spalte idfrei darf nicht explizit gesetzt werden
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['idfrei'], ARRAY['0'] );

-- Der Wert war bereits in der Datenbank enthalten
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['abczk'], ARRAY['KeinRichtigesWort']
);
SELECT * FROM intern.abczk ORDER BY id DESC LIMIT 5;
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['abczk'], ARRAY['KeinRichtigesWort']
);
SELECT * FROM intern.abczk ORDER BY id DESC LIMIT 5;

-- Wiederverwendung einer gelöschten id
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['abczk'], ARRAY['KeinRichtigesWort']
);
```

```
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['abczk'], ARRAY['KeinRichtigesWortZwe
);
SELECT * FROM intern.abczk ORDER BY id DESC LIMIT 5;
SELECT intern.loesche__zeile( 'intern.abczk', 'abczk = ''KeinRichtigesWort''' );
SELECT * FROM intern.abczk ORDER BY id DESC LIMIT 5;
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['abczk'], ARRAY['KeinRichtigesWortEin
);
SELECT * FROM intern.abczk ORDER BY id DESC LIMIT 5;

-- Überschreiben einer Zeile mit vorgegebener id
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['abczk'], ARRAY['KeinRichtigesWort']
);
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['abczk'], ARRAY['KeinRichtigesWortZwe
);
SELECT * FROM intern.abczk ORDER BY id DESC LIMIT 5;
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['id', 'abczk'], ARRAY['2399',
'KeinRichtigesWortEins'] );
SELECT * FROM intern.abczk ORDER BY id DESC LIMIT 5;

-- Einfügen einer Zeile mit vorgegebener id
SELECT * FROM intern.abczk ORDER BY id DESC LIMIT 5;
SELECT intern.nimm__zeile( 'intern.abczk', ARRAY['id', 'abczk'], ARRAY['999999',
'NichtVorhandeneId'] );
SELECT * FROM intern.abczk ORDER BY id DESC LIMIT 5;
```

4.21 intern.pruefe__array

Funktionsname

`intern.pruefe__array( IN liste varchar[], IN minel int4, IN maxel int4, IN fktname varchar, IN paramname varchar )`

Funktionsparameter/Rückgabewert(e)

`liste` Es ist die Liste anzugeben, deren Plausibiliät geprüft werden soll.

`minel` Es ist die Anzahl der Listenelemente anzugeben, welche mindestens übergeben werden sollen. Wenn kleiner `1` oder `NULL` übergeben wird, muß die Liste `NULL` sein.

`maxel` Es ist die Anzahl der Listenelemente anzugeben, welche maximal übergeben werden sollen. Wenn kleiner `1` oder `NULL` übergeben wird, darf die Liste eine beliebige Anzahl Elemente besitzen. Andernfalls darf die Anzahl der Elemente den übergebenen Wert nicht überschreiten.

`fktname` Es ist der Name der Funktion anzugeben, welche die Prüfroutine aufruft. Der Funktionsname wird in der Fehlermeldung verwendet. Sollte er leer oder `NULL` sein, wird `intern.pruefe__array(...)` verwendet.

`paramname` Es ist der Name des Parameters anzugeben, welcher die zu prüfende Liste `liste` beinhaltet. Der Parametername wird in der Fehlermeldung verwendet. Sollte er leer oder `NULL` sein, wird `liste` verwendet.

Funktionsbeschreibung
Ziel der Funktion ist die Plausibilitätsprüfung der mittels des Parameters `liste` übergebenen Liste (`ARRAY`) anhand der Randbedingungen `minel` und `maxel`. Wenn die Liste nicht `NULL` sein darf, darf keines der Listenelemente leer sein. Sollte die Plausibilitätsprüfung negativ ausfallen, wird eine Fehlermeldung ausgegeben.

Beispiele

```
Fehlermeldung mit Standardfunktions- und Parameternamen ausgeben, wenn liste nicht
NULL sein darf.
SELECT * FROM intern.pruefe__array( NULL, 1, 1, NULL, NULL );
SELECT * FROM intern.pruefe__array( NULL, 1, 1, '', '' );

Fehlermeldung mit übergebenen Funktions- und Parameternamen ausgeben, wenn liste
nicht NULL sein darf.
SELECT * FROM intern.pruefe__array( NULL, 1, 1, 'fktname(...)', 'listenName' );

Fehlermeldung mit Standardfunktions- und Parameternamen ausgeben, wenn liste NULL
sein muß.
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins', 'zwei' ], NULL, 1, NULL, NULL
);
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins', 'zwei' ], 0, 1, '', '' );
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins', 'zwei' ], -5, 1, NULL, NULL
);

Fehlermeldung mit übergebenen Funktions- und Parameternamen ausgeben, wenn liste
NULL sein muß.
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins', 'zwei' ], NULL, 1, 'fktname(...)',
'listenName' );

Fehlermeldung ausgeben, wenn liste mindesten 2 Elemente enthalten muß.
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins' ], 2, 0, NULL, NULL );
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins' ], 2, NULL, 'fktname(...)', 'listenName'
);

Fehlermeldung ausgeben, wenn liste maximal 1 Element enthalten darf.
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins', 'zwei' ], 1, 1, '', '' );
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins', 'zwei' ], 1, 1, 'fktname(...)',
'listenName' );

Fehlermeldung ausgeben, wenn liste maximal 1 Element enthalten darf.
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins', '', 'drei' ], 1, 3, NULL, NULL
);
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins', '', 'drei' ], 1, 3, '', '' );
```

```
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins', '', 'drei' ], 1, 3, 'fktname(...)',
'listenName' );

keine Fehlermeldung ausgeben, weil liste NULL sein darf.
SELECT * FROM intern.pruefe__array( NULL, NULL, 1, 'fktname(...)', 'listenName'
);
SELECT * FROM intern.pruefe__array( NULL, 0, 1, 'fktname(...)', 'listenName' );
SELECT * FROM intern.pruefe__array( NULL, -5, 1, 'fktname(...)', 'listenName' );

keine Fehlermeldung ausgeben, weil liste minimal 1 und maximal 2 Element enthalten
darf.
SELECT * FROM intern.pruefe__array( ARRAY[ 'eins', 'zwei' ], 1, 2, 'fktname(...)',
'listenName' );
```

4.22 intern.pruefe__dt

Funktionsname
`intern.pruefe__dt( IN dt varchar, IN wert varchar, IN ug varchar, IN og varchar, IN fktname varchar, IN paramname varchar )`

Funktionsparameter/Rückgabewert(e)

`dt` Es ist einer der möglichen Datentypen (siehe Funktionsbeschreibung — Gültige Datentypen / Wertebereich) anzugeben

`wert` Es ist der Wert anzugeben, dessen Plausibiliät geprüft werden soll. Er darf nicht `NULL` oder leer sein.

`ug` Es ist die untere Grenze des Wertes anzugeben (weitere Hinweise siehe Funktionsbeschreibung — Gültige Datentypen / Wertebereich). Wird `NULL` oder eine Leerzeichenkette angegeben, gilt die in PostgreSQL implementierte untere Grenze.

`og` Es ist die obere Grenze des Wertes anzugeben. Wird `NULL` oder eine Leerzeichenkette angegeben, gilt die in PostgreSQL implementierte obere Grenze.

`fktname` Es ist der Name der Funktion anzugeben, welche die Prüfroutine aufruft. Der Funktionsname wird in der Fehlermeldung verwendet. Sollte er leer oder `NULL` sein, wird `intern.pruefe__dt(...)` verwendet.

`paramname` Es ist der Name des Parameters anzugeben, welcher den zu prüfenden Wert `wert` beinhaltet. Der Parametername wird in der Fehlermeldung verwendet. Sollte er leer oder `NULL` sein, wird `wert` verwendet.

Funktionsbeschreibung
Ziel der Funktion ist die Plausibilitätsprüfung elementarer Datentypen. Die Datentypen und der Wertebereich werden als Zeichenketten übergeben. Sollte die Plausibilitätsprüfung negativ

ausfallen, wird eine Fehlermeldung ausgegeben.
Der im PostgreSQL implementierte Wertebereich der Datentypen kann der Postgres-Dokumentation entnommen werden. Er wird von PostgreSQL automatisch und somit innerhalb dieser Rputine nicht explizit überprüft.

Gültige Datentypen / Wertebereich
Es werden die folgenden Datentypen geprüft:

bw Boolescher Wert. Die untere / obere Grenze wird nicht überprüft.

du Datum / Uhrzeit. Die untere / obere Grenze wird überprüft.

gz Ganzzahl (integer). Die untere / obere Grenze wird überprüft.

zk Zeichenkette. Die untere Grenze wird nicht überprüft. Die obere Grenze wird überprüft.

Beispiele

```
Fehlermeldung. Der übergebene Datentyp darf nicht NULL oder leer sein
SELECT * FROM intern.pruefe_dt( NULL, '1', NULL, NULL, NULL, NULL );

Fehlermeldung. Der übergebene Datentyp ist unbekannt
SELECT * FROM intern.pruefe_dt( 'unbekannt', '1', NULL, NULL, 'fktname(...)',
'wertName' );

Fehlermeldung: Es ist kein boolean
SELECT * FROM intern.pruefe_dt( 'bw', 'keinBool', 'ignoriert', 'ignoriert', NULL,
NULL );
SELECT * FROM intern.pruefe_dt( 'bw', 'keinBool', 'ignoriert', 'ignoriert', 'fktname(...)
'wertName' );

Fehlermeldung. Der übergebene Wert darf nicht NULL oder leer sein
SELECT * FROM intern.pruefe_dt( NULL, NULL, NULL, NULL, NULL, NULL );
SELECT * FROM intern.pruefe_dt( NULL, '', NULL, NULL, 'fktname(...)', 'wertName'
);

Fehlermeldung. Das Datum befindet sich außerhalb der zulässigen Grenzen
SELECT * FROM intern.pruefe_dt( 'du', '2007-04-16', '2007-04-17', NULL, 'fktname(...)',
'datUhrz' );
SELECT * FROM intern.pruefe_dt( 'du', '2007-04-18', NULL, '2007-04-17', 'fktname(...)',
'datUhrz' );
SELECT * FROM intern.pruefe_dt( 'du', '2007-04-19', '2007-04-17', '2007-04-18',
'fktname(...)', 'datUhrz' );

Fehlermeldung. Die Ganzzahl (integer) befindet sich außerhalb der zulässigen Grenzen
SELECT * FROM intern.pruefe_dt( 'gz', '-2', '-1', NULL, 'fktname(...)', 'gz_int'
);
```

```
SELECT * FROM intern.pruefe_dt( 'gz', '2', NULL, '1', 'fktname(...)', 'gz_int'
);
SELECT * FROM intern.pruefe_dt( 'gz', '-5', '-1', '1', 'fktname(...)', 'gz_int'
);

Fehlermeldung. Die übergebene Zeichenkette ist zu lang
SELECT * FROM intern.pruefe_dt( 'zk', '1234567890', NULL, '9', NULL, NULL );
SELECT * FROM intern.pruefe_dt( 'zk', '1234567890', NULL, '9', 'fktname(...)',
'zkWert' );

keine Fehlermeldung
SELECT * FROM intern.pruefe_dt( 'bw', 'TRUE', NULL, NULL, 'fktname(...)', 'wertName'
);
SELECT * FROM intern.pruefe_dt( 'bw', 'T', NULL, NULL, 'fktname(...)', 'wertName'
);
SELECT * FROM intern.pruefe_dt( 'bw', 'Y', NULL, NULL, 'fktname(...)', 'wertName'
);
SELECT * FROM intern.pruefe_dt( 'bw', 'YES', NULL, NULL, 'fktname(...)', 'wertName'
);
SELECT * FROM intern.pruefe_dt( 'bw', '1', NULL, NULL, 'fktname(...)', 'wertName'
);
SELECT * FROM intern.pruefe_dt( 'bw', 'FALSE', NULL, NULL, 'fktname(...)', 'wertName'
);
SELECT * FROM intern.pruefe_dt( 'bw', 'F', NULL, NULL, 'fktname(...)', 'wertName'
);
SELECT * FROM intern.pruefe_dt( 'bw', 'N', NULL, NULL, 'fktname(...)', 'wertName'
);
SELECT * FROM intern.pruefe_dt( 'bw', 'NO', NULL, NULL, 'fktname(...)', 'wertName'
);
SELECT * FROM intern.pruefe_dt( 'bw', '0', NULL, NULL, 'fktname(...)', 'wertName'
);
SELECT * FROM intern.pruefe_dt( 'du', '2007-04-18', NULL, NULL, 'fktname(...)',
'wertName' );
SELECT * FROM intern.pruefe_dt( 'du', '2007-04-18', '2007-04-17', '2007-04-19',
'fktname(...)', 'wertName' );
SELECT * FROM intern.pruefe_dt( 'gz', '2123456', NULL, NULL, 'fktname(...)', 'wertNam
);
SELECT * FROM intern.pruefe_dt( 'gz', '0', '-1', '1', 'fktname(...)', 'wertName'
);
SELECT * FROM intern.pruefe_dt( 'zk', '123456789', 'ignoriert', '9', 'fktname(...)',
'wertName' );
```

4.23 intern.verbindezk

Funktionsname
`intern.verbindezk( IN zk varchar, IN zkadd varchar, OUT zkges varchar )`

Funktionsparameter/Rückgabewert(e)

`zk` Erste Zeichenkette.

`zkadd` Zweite Zeichenkette.

`zkges` Erste + Zweite Zeichenkette.

Funktionsbeschreibung
Ziel der Funktion ist das Verbinden beider Zeichenketten. Diese Funktion wird in der Aggregatfunktion `intern.erzeuge_zk(...)` verwendet.

Beispiele
```
intern.verbindezk( 'Anfang ', '- Ende' );
```

4.24 intern.zeige_fremdschluessel

Beschreibung der Spalten

`fk` Namen der Fremdschlussel in der Datenbank

`spstart` Schema.Tabelle.Name der Startspalte des Fremdschlüssels

`spziel` Schema.Tabelle.Name der Zielspalte des Fremdschlüssels

Beschreibung der Sicht
Es werden alle im System vorhandenen Fremdschlüssel einschließlich der Start- und Zielspalten aufgelistet.

4.25 intern.zeige_spaltenanzahl

Beschreibung der Spalten

`tabname` Tabellenname

`spanz` Anzahl der Spalten der Tabelle

Beschreibung der Sicht
Es wird die Anzahl der in der Tabelle enthaltenen Spalten (ohne Systemspalten) aufgelistet.

4.26 intern.zeige__tabspinformationen

Beschreibung der Spalten

`spname` Schema.Tabelle.Name der Spalte

`dtyp` pg_type.typname

`spnr` pg_attribute.attnum

`beschr_spalte` pg_description.description für die Spalte

`beschr_tabelle` pg_description.description für die Tabelle

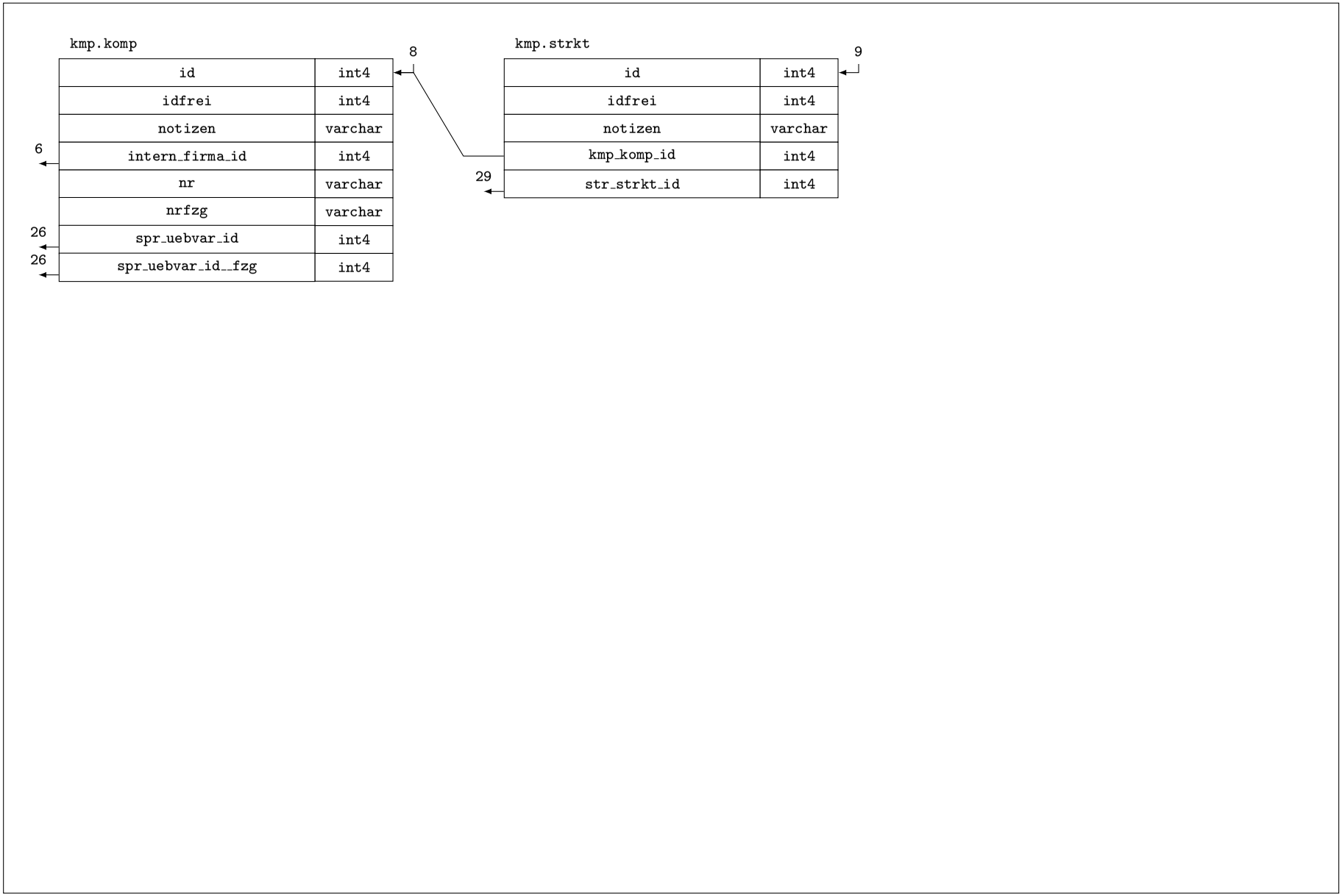
Beschreibung der Sicht

Es werden alle im System vorhandenen Spalten (einschließlich Schema- und Tabellennamen) und deren Beschreibung aufgelistet.

5 kmp

Im Schema „Komponente“ werden die technischen Sytsteme / Komponenten gehalten und Strukturaspekte zugeordnet.

5.1 Tabellenhierarchie der Ebenen 1 — 3



5.2 Schlüssel und Bedingungen

Primärschlüssel

pk__kmp_komp_id für Spalte kmp.komp.id

pk__kmp_strkt_id für Spalte kmp.strkt.id

Fremdschlüssel

fk__kmp_komp__intern_firma_id aus Spalte kmp.komp.intern_firma_id

fk__kmp_komp__spr_uebvar_id aus Spalte kmp.komp.spr_uebvar_id

fk__kmp_komp__spr_uebvar_id__fzg aus Spalte kmp.komp.spr_uebvar_id__fzg

fk__kmp_strkt__kmp_komp_id aus Spalte kmp.strkt.kmp_komp_id

fk__kmp_strkt__str_strkt_id aus Spalte kmp.strkt.str_strkt_id

Unique Constraints

uc__kmp_komp in Tabelle kmp.komp für die Spalte(n) kmp.komp.idfrei, kmp.komp.intern_firma_id, kmp.komp.nr

uc__kmp_strkt in Tabelle kmp.strkt für die Spalte(n) kmp.strkt.idfrei, kmp.strkt.kmp_komp_id, kmp.strkt.str_strkt_id

5.3 kmp.nimm__komp

Funktionsname

```
kmp.nimm__komp( IN liste_werte varchar[], OUT kompid varchar )
```

Funktionsparameter/Rückgabewert(e)

liste_werte Liste der Werte, welche in kmp.komp eingelesen werden sollen (Hinweise siehe Funktionsbeschreibung — Werteliste einlesen). Sie werden in der folgenden Reihenfolge in die Spalten

1. intern_firma_id
2. nr
3. spr_uebvar_id
4. nrfzg
5. spr_uebvar_id__fzg
6. notizen

eingetragen. Es müssen mindestens die ersten zwei Werte übergeben werden.

kompid Es wird die Datenbank-Identifikationsnummer der eingelesenen Komponente (kmp.komp.id) zurückgegeben.

Funktionsbeschreibung
Mittels dieser Funktion werden die im System verwendbaren Komponenten in `kmp.komp` eingelesen.

Werteliste einlesen:
Dem Funktionsparameter `liste_werte` ist eine Werteliste (`ARRAY`) zu übergeben (die maximale Anzahl und die Reihenfolge der Werte ist fest vorgegeben — siehe oben). Soll nur eine Teilmenge der Werte übergeben werden, dann sind alle Werte von links nach rechts zu spezifizieren (nichtbenötigte Werte am Ende der Liste dürfen weggelassen werden).

Beispiele
```
-- Fehlermeldung. Der Parameter liste_werte darf nicht NULL sein.
SELECT * FROM kmp.nimm__komp( NULL );

-- Fehlermeldung. Es sind mindestens zwei Werte zu übergeben.
SELECT * FROM kmp.nimm__komp( ARRAY['1'] );

-- Fehlermeldung. Die Werte müssen größer 0 bzw. dürfen nicht leer sein.
SELECT * FROM kmp.nimm__komp( ARRAY['0', '08-15'] );
SELECT * FROM kmp.nimm__komp( ARRAY['1', ''] );

-- Komponente einlesen
SELECT kmp_komp_id, intern_firma_id__herst, kmpnr, spr_uebvar_id__herstbez, kmpnrfzgherst
spr_uebvar_id__fzgherstbez, kmp_komp_notizen FROM kmp.zeige__komp;
SELECT * FROM kmp.nimm__komp( ARRAY['1', '08-15', spr.nimm__wgruppe( 'Komponentenbezeic
'1', '0' )] );
SELECT * FROM kmp.nimm__komp( ARRAY['1', '08-15', spr.nimm__wgruppe( 'Komponentenbezeic
'1', '0' ), 'fzg-08-15'] );
SELECT * FROM kmp.nimm__komp( ARRAY['1', '08-15', spr.nimm__wgruppe( 'Komponentenbezeic
'1', '0' ), 'fzg-08-15', spr.nimm__wgruppe( 'Komponentenbezeichnung-Fahrezeugherstelle
'1', '0' )] );
SELECT * FROM kmp.nimm__komp( ARRAY['1', '08-15', spr.nimm__wgruppe( 'Komponentenbezeic
'1', '0' ), 'fzg-08-15', spr.nimm__wgruppe( 'Komponentenbezeichnung-Fahrezeugherstelle
'1', '0' ), 'Zusatzinfo'] );
SELECT kmp_komp_id, intern_firma_id__herst, kmpnr, spr_uebvar_id__herstbez, kmpnrfzgherst
spr_uebvar_id__fzgherstbez, kmp_komp_notizen FROM kmp.zeige__komp;
```

5.4 kmp.nimm__strkt

Funktionsname
`kmp.nimm__strkt( IN tabwerte varchar[], OUT strktid varchar )`

Funktionsparameter/Rückgabewert(e)

`tabwerte` Liste der Werte, welche in `kmp.strkt` eingelesen werden sollen (Hinweise siehe

Funktionsbeschreibung — Werteliste einlesen). Sie werden in der folgenden Reihenfolge in die Spalten

1. `kmp_komp_id`
2. `str_strkt_id`
3. `notizen`

eingetragen. Es müssen mindestens die ersten beiden Werte übergeben werden.

`strktid` Es wird die Identifikationsnummer der aktuell eingelesenen Strukturpunktzuordnung `kmp.strkt.id` zurückgegeben.

Funktionsbeschreibung
Mittels dieser Funktion werden der aktuellen Komponente die Strukturpunkte zugeordnet (in `kmp.strkt` eingelesen).

Werteliste einlesen:
Dem Funktionsparameter `tabwerte` ist eine Werteliste (`ARRAY`) zu übergeben (die maximale Anzahl und die Reihenfolge der Werte ist fest vorgegeben — siehe oben). Soll nur eine Teilmenge der Werte übergeben werden, dann sind alle Werte von links nach rechts zu spezifizieren (nichtbenötigte Werte am Ende der Liste dürfen weggelassen werden).

Beispiele
```
-- Fehlermeldung. Es müssen mindestens zwei Elemente übergeben werden
SELECT * FROM kmp.nimm__strkt( ARRAY['1'] );

-- Fehlermeldung. Die übergebenen Parameter müssen größer 0 sein.
SELECT * FROM kmp.nimm__strkt( ARRAY['0', '1'] );
SELECT * FROM kmp.nimm__strkt( ARRAY['1', '0'] );

-- Strukturverbindung einlesen
SELECT * FROM kmp.strkt;
SELECT * FROM kmp.nimm__strkt( ARRAY[kmp.nimm__komp( ARRAY['1', '08-15'] ), str.nimm__strkt(
ARRAY['1', '-BAAA', spr.nimm__wgruppe( 'Strukturpunkt', '1', '0' )] ), 'Zusatzinformationen
);
SELECT * FROM kmp.strkt;
```

5.5 kmp.zeige__komp

Beschreibung der Spalten

`kmp_komp_id` siehe `kmp.komp.id`

`intern_firma_id__herst` siehe `kmp.komp.intern_firma_id`

`kmpnr` siehe `kmp.komp.nr`

spr_uebvar_id__herstbez siehe kmp.komp.spr_uebvar_id

kmpnrfzgherst siehe kmp.komp.nrfzg

spr_uebvar_id__fzgherstbez siehe kmp.komp.spr_uebvar_id__fzg

kmp_komp_notizen siehe kmp.komp.notizen

kmp_strkt_id siehe kmp.strkt.id

str_strkt_id siehe kmp.strkt.str_strkt_id

kmp_strkt_notizen siehe kmp.strkt.notizen

Beschreibung der Sicht
Es werden die verfügbaren Komponenten einschließlich zugeordneter Strukturaspekte zurückgeben.

6 prj

Ausschließlich im Schema „Projekte“ wird der Projektbezug für alle weiteren Schemas hergestellt. Die Definition des Begriffes „Projekt“ hängt selbstverständlich von der Anwendung ab.

In der vorliegenden Datenbank kann mittels des Projektes „DBDokumentation“ die Dokumentation der Datenbank aus den Kommentaren der verschiedenen Datenbankobjekte erzeugt werden.
Dazu sind in der Kommandozeile (Eingabeaufforderung engl. Shell) die folgenden Befehle einzugeben:

1. `psql -U micha vf -P tuples_only -P format=unaligned -P footer -c ‘‘SELECT replace( makefile, ’-U micha vf’, ’-U ’ || current_user || ’ ’ || current_database() ) FROM prj.projekte WHERE prjname = ’DBDokumentation’;‘‘ -o makefile`
   Anstelle des Benutzers `micha` im shell-Befehl `psql -U micha vf ...` und des Datenbanknamens `vf` ist der eigene Benutzername bzw. Datenbankname anzugeben

2. `make`

3. Abschließend ist die erzeugte pdf-Datei zu öffnen.

Das genannte Verfahren kann für sämtliche Projekte, welchen ein `makefile` zugeordnet ist analog angewendet werden. Mögliche Varianten zum erzeugen unterschiedlicher Dokumente können dem `makefile` des entsprechenden Projektes entnommen werden.

Projektbezug

1. Projektzuordnung

2. Zuordung von Personen

3. Logbuchfunktion

4. Umbenennen von Standardvorgaben

	1	2	3	4
dok.ausgabe	x	x	-	-
dok.doku	-	-	-	-
dok.komp	x	-	-	-
dok.strkt	x	-	-	-
kmp.komp	x	x	-	x
kmp.strkt	x	-	-	-
prt.dokausgabe	x	-	-	-
prt.komp	x	-	-	-
prt.protokoll	x	x	-	x
prt.strkt	x	-	-	-
prt.verbindung	x	-	-	-
ps.abk	x	-	-	x
ps.gefabk	x	-	x	-
ps.gefanm	x	-	x	x
ps.gefkomp	x	-	x	-
ps.gefnetz	x	-	x	-
ps.gefprot	x	-	x	-
ps.gefstrkt	x	-	x	-
ps.gefszenario	x	x	-	x
str.name	-	-	-	-
str.strkt	x	x	-	x
str.verbindung	x	-	-	x

Der Projektbezug muß explizit in der gesamten Abfragekette bis zur letzten Ebene hergestellt werden.

Benennen der Tabellen

`prj.p_*` Projektbezug (1)

`prj.pp_*` projektbezogene Personenzuordnung (1 + 2)

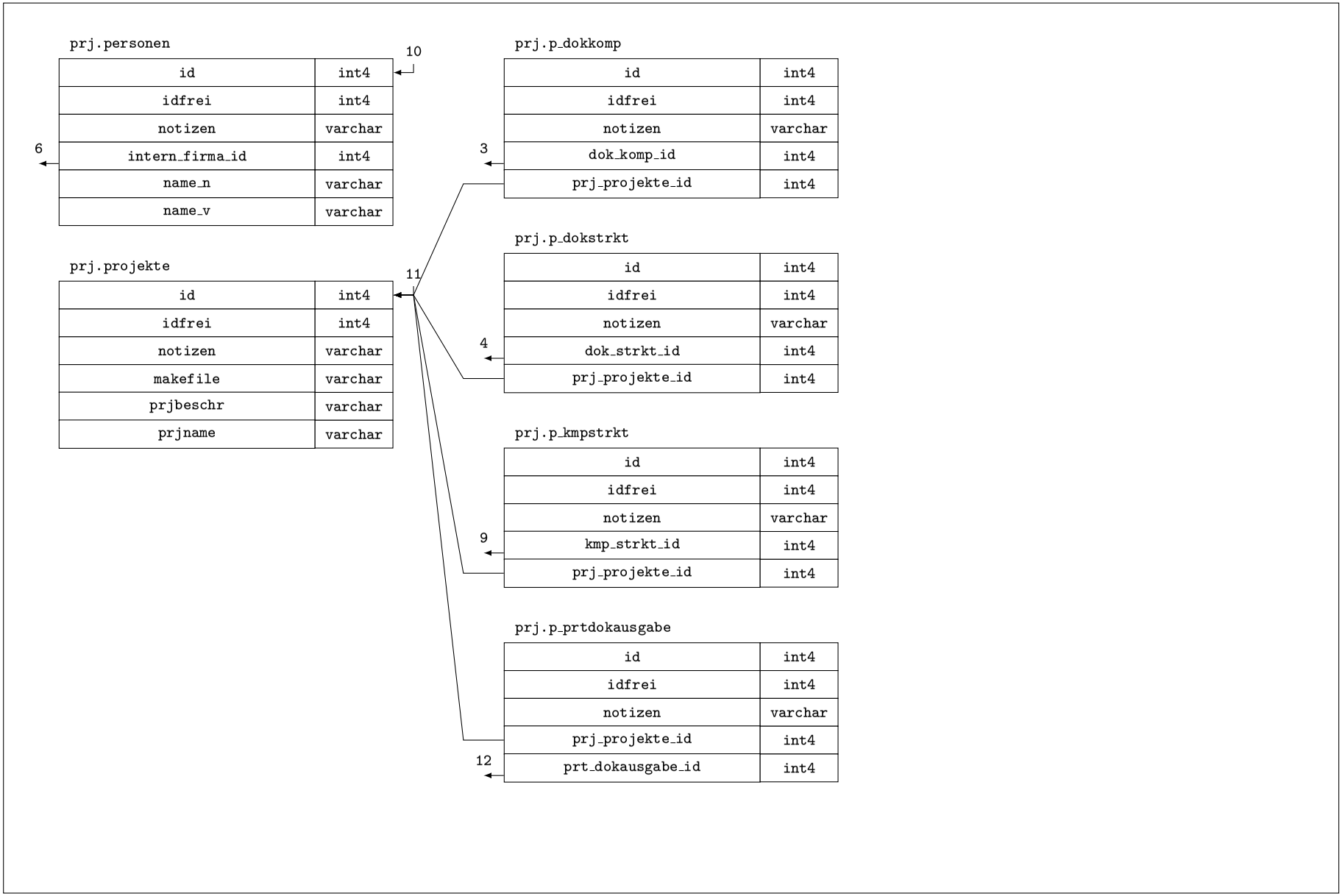
`prj.ppu_*` projektbezogene Personenzuordnung und Umbenennen von Texten (1 + 2 + 4)

`prj.pl_*` projektbezogene Logbuchfunktionalität (1 + 3)

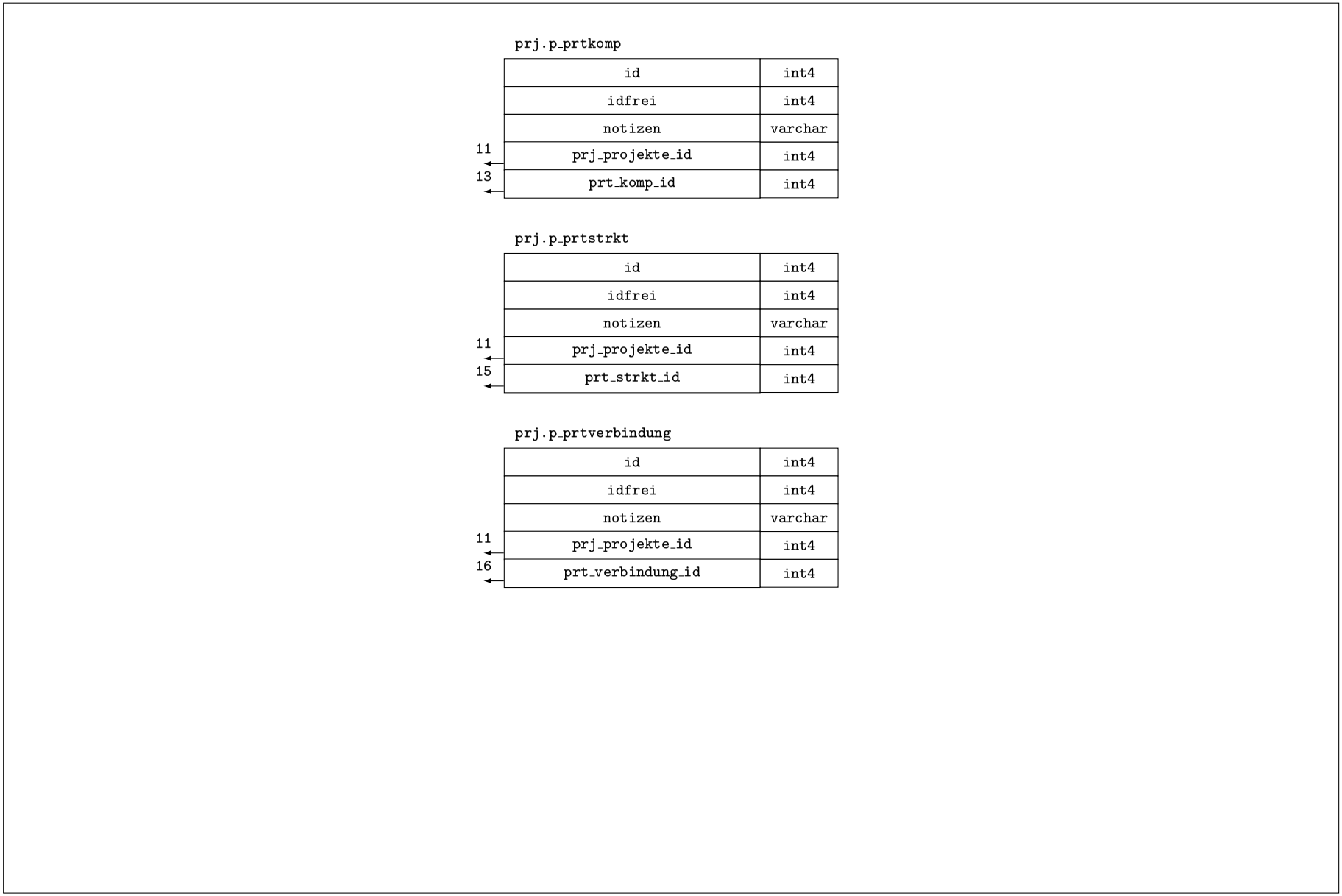
`prj.plu_*` projektbezogene Logbuchfunktionalität und Umbenennen von Texten (1 + 3 + 4)

`prj.pu_*` projektbezogenes Umbenennen von Texten (1 + 4)

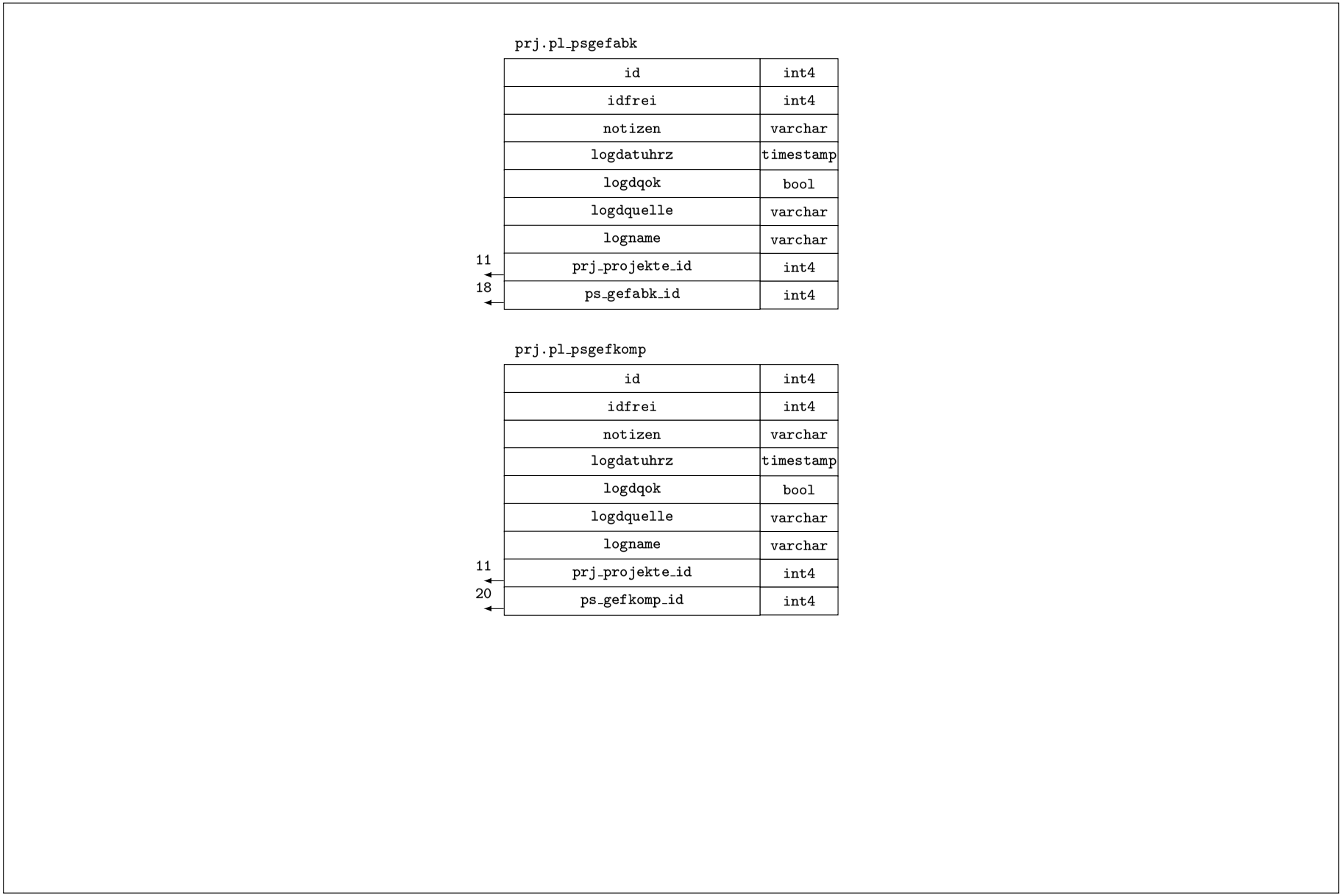
6.1 Tabellenhierarchie der Ebenen 1 — 3



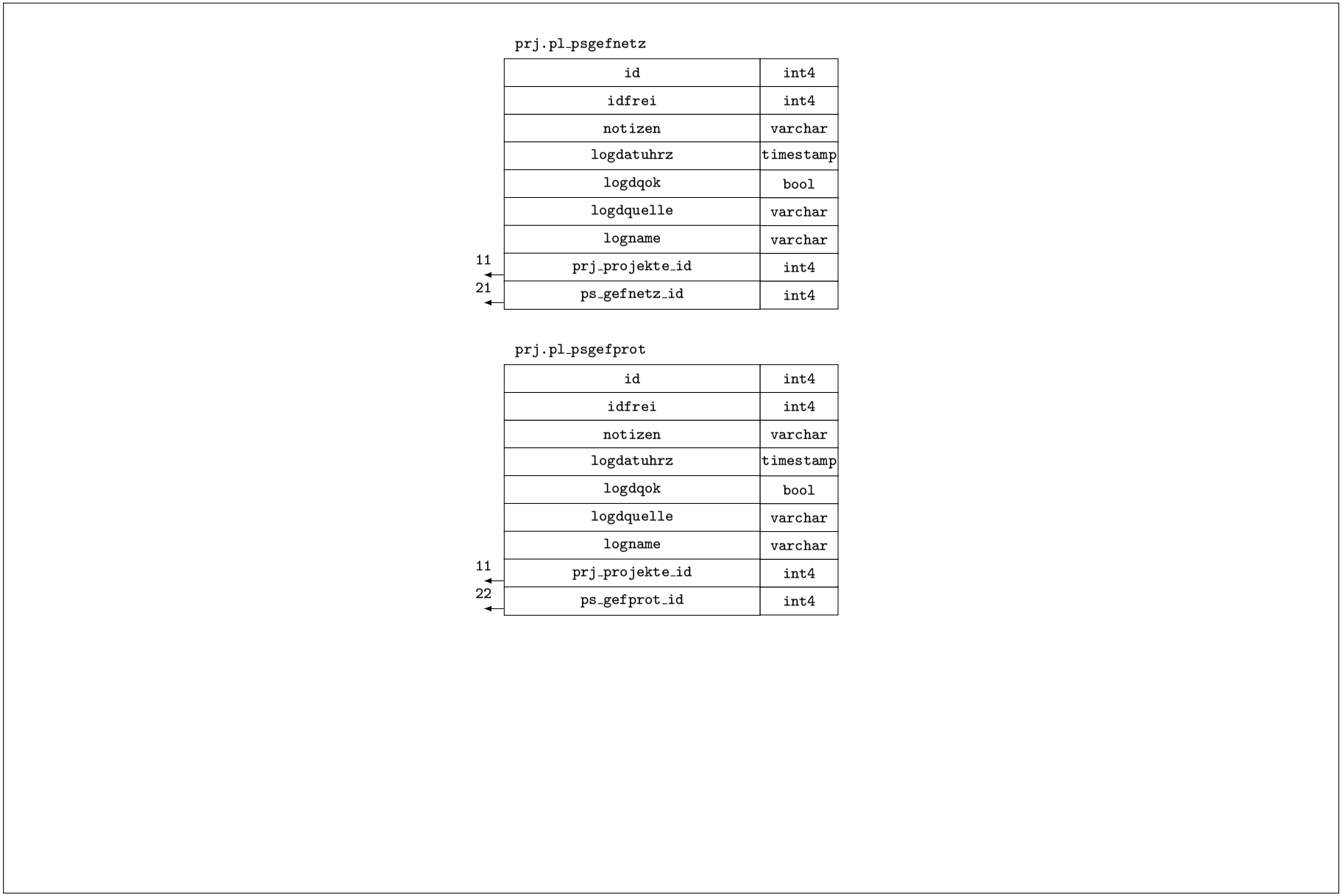
6.2 Tabellenhierarchie der Ebenen 1 — 3



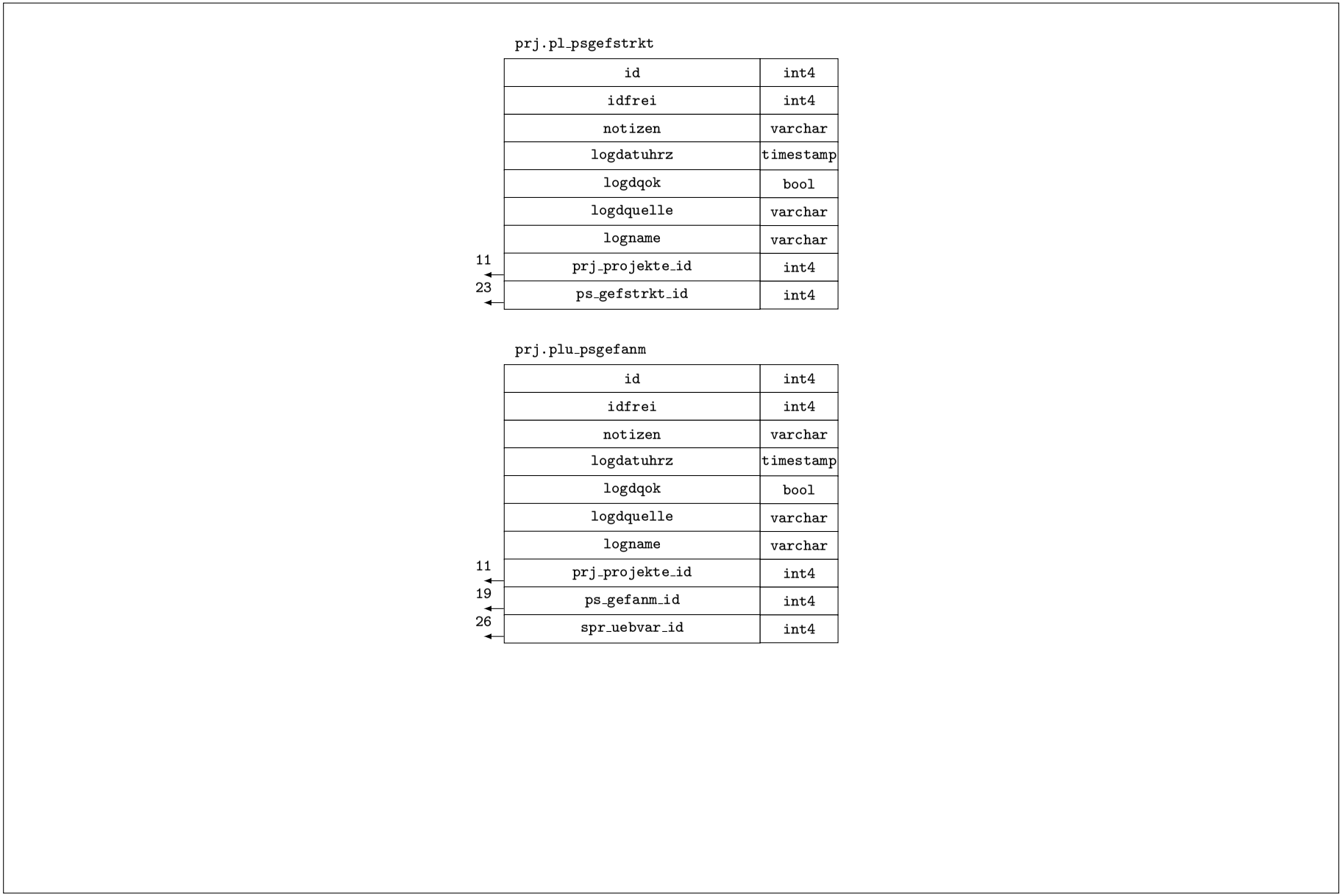
6.3 Tabellenhierarchie der Ebenen 1 — 3



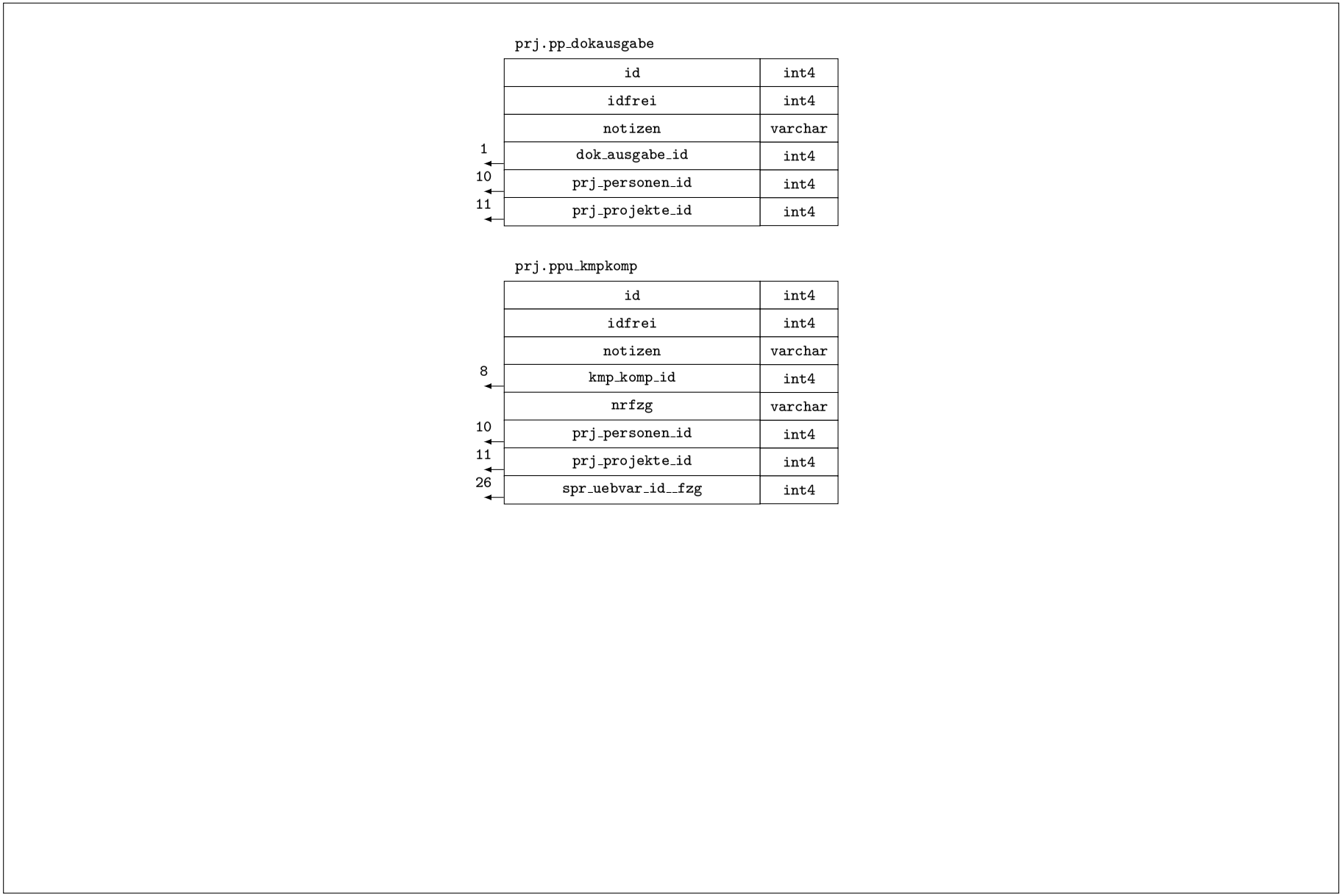
6.4 Tabellenhierarchie der Ebenen 1 — 3



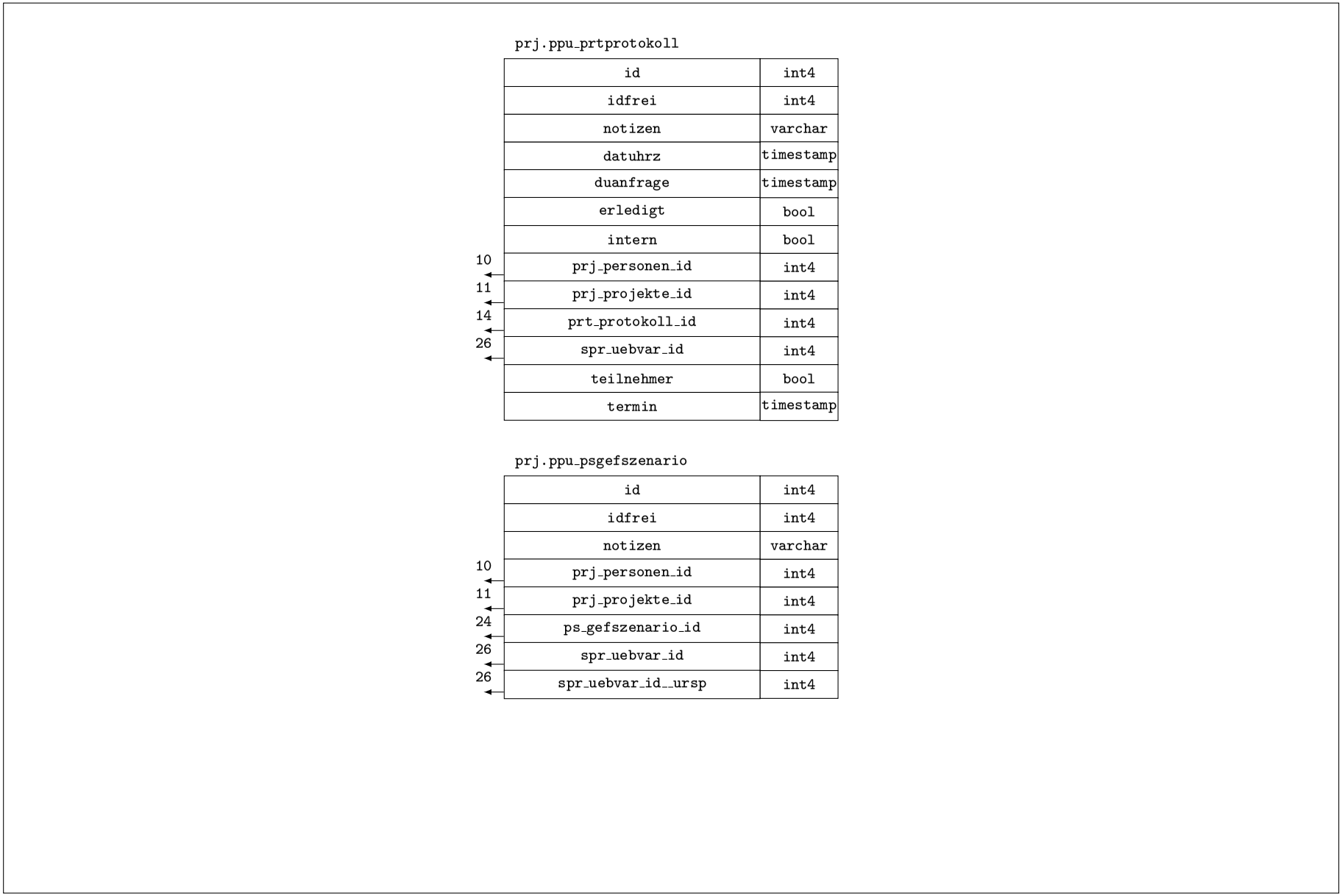
6.5 Tabellenhierarchie der Ebenen 1 — 3



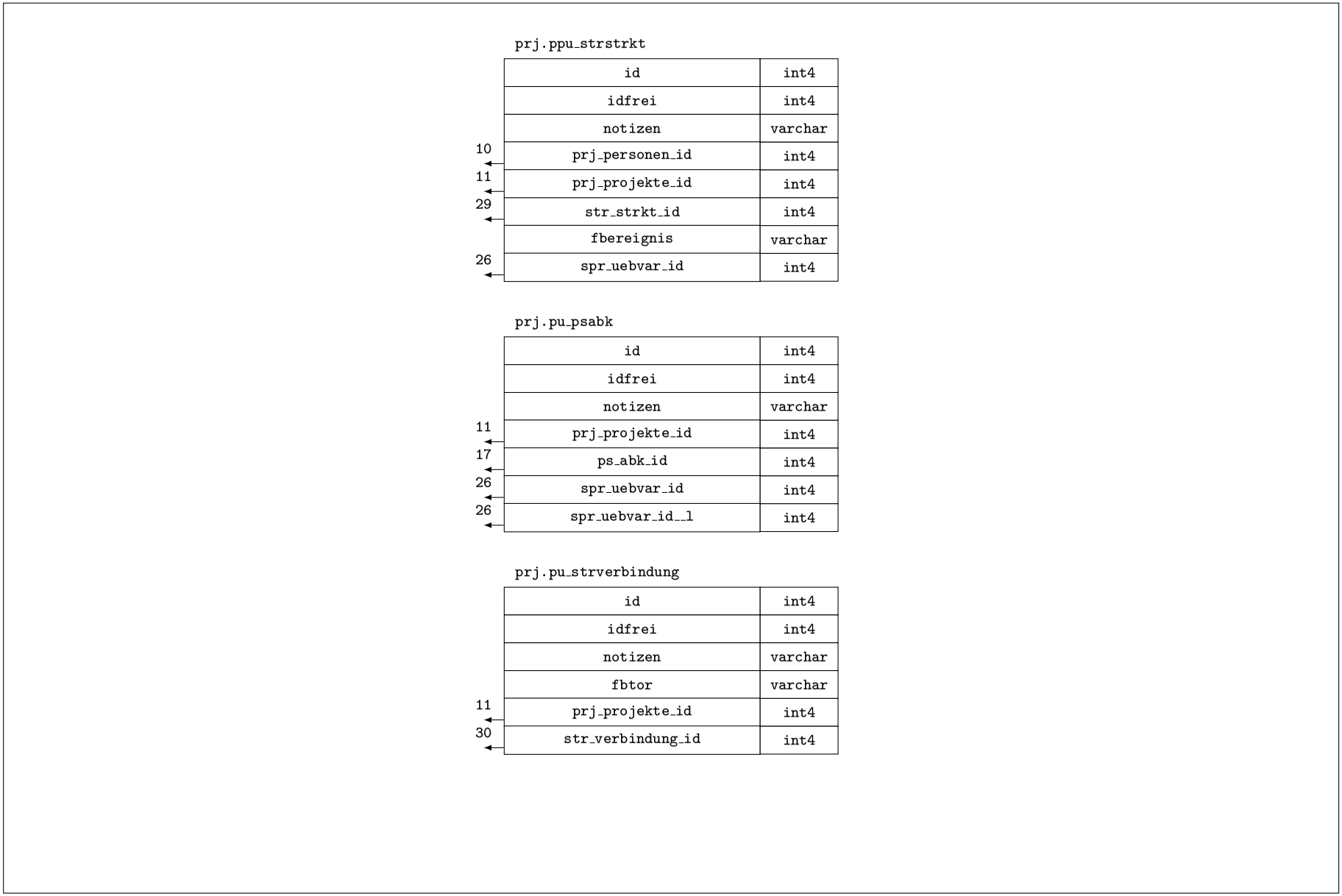
6.6 Tabellenhierarchie der Ebenen 1 — 3



6.7 Tabellenhierarchie der Ebenen 1 — 3



6.8 Tabellenhierarchie der Ebenen 1 — 3



6.9 Schlüssel und Bedingungen

Primärschlüssel

pk__prj_p_dokkomp_id für Spalte prj.p_dokkomp.id

pk__prj_p_dokstrkt_id für Spalte prj.p_dokstrkt.id

pk__prj_p_kmpstrkt_id für Spalte prj.p_kmpstrkt.id

pk__prj_p_prtdokausgabe_id für Spalte prj.p_prtdokausgabe.id

pk__prj_p_prtkomp_id für Spalte prj.p_prtkomp.id

pk__prj_p_prtstrkt_id für Spalte prj.p_prtstrkt.id

pk__prj_p_prtverbindung_id für Spalte prj.p_prtverbindung.id

pk__prj_personen_id für Spalte prj.personen.id

pk__prj_pl_psgefabk_id für Spalte prj.pl_psgefabk.id

pk__prj_pl_psgefkomp_id für Spalte prj.pl_psgefkomp.id

pk__prj_pl_psgefnetz_id für Spalte prj.pl_psgefnetz.id

pk__prj_pl_psgefprot_id für Spalte prj.pl_psgefprot.id

pk__prj_pl_psgefstrkt_id für Spalte prj.pl_psgefstrkt.id

pk__prj_plu_psgefanm_id für Spalte prj.plu_psgefanm.id

pk__prj_pp_dokausgabe_id für Spalte prj.pp_dokausgabe.id

pk__prj_ppu_kmpkomp_id für Spalte prj.ppu_kmpkomp.id

pk__prj_ppu_prtprotokoll_id für Spalte prj.ppu_prtprotokoll.id

pk__prj_ppu_psgefszenario_id für Spalte prj.ppu_psgefszenario.id

pk__prj_ppu_strstrkt_id für Spalte prj.ppu_strstrkt.id

pk__prj_projekte_id für Spalte prj.projekte.id

pk__prj_pu_psabk_id für Spalte prj.pu_psabk.id

pk__prj_pu_strverbindung_id für Spalte prj.pu_strverbindung.id

Fremdschlüssel

fk__prj_p_dokkomp__dok_komp_id aus Spalte prj.p_dokkomp.dok_komp_id

fk__prj_p_dokkomp__prj_projekte_id aus Spalte prj.p_dokkomp.prj_projekte_id

fk__prj_p_dokstrkt__dok_strkt_id aus Spalte prj.p_dokstrkt.dok_strkt_id

fk__prj_p_dokstrkt__prj_projekte_id aus Spalte prj.p_dokstrkt.prj_projekte_id

fk__prj_p_kmpstrkt__kmp_strkt_id aus Spalte prj.p_kmpstrkt.kmp_strkt_id

fk__prj_p_kmpstrkt__prj_projekte_id aus Spalte prj.p_kmpstrkt.prj_projekte_id

fk__prj_p_prtdokausgabe__prj_projekte_id aus Spalte prj.p_prtdokausgabe.prj_projekte_id

fk__prj_p_prtdokausgabe__prt_dokausgabe_id aus Spalte prj.p_prtdokausgabe.prt_dokausgabe_id

fk__prj_p_prtkomp__prj_projekte_id aus Spalte prj.p_prtkomp.prj_projekte_id

fk__prj_p_prtkomp__prt_komp_id aus Spalte prj.p_prtkomp.prt_komp_id

fk__prj_p_prtstrkt__prj_projekte_id aus Spalte prj.p_prtstrkt.prj_projekte_id

fk__prj_p_prtstrkt__prt_strkt_id aus Spalte prj.p_prtstrkt.prt_strkt_id

fk__prj_p_prtverbindung__prj_projekte_id aus Spalte prj.p_prtverbindung.prj_projekte_id

fk__prj_p_prtverbindung__prt_verbindung_id aus Spalte prj.p_prtverbindung.prt_verbindung_id

fk__prj_personen__intern_firma_id aus Spalte prj.personen.intern_firma_id

fk__prj_pl_psgefabk__prj_projekte_id aus Spalte prj.pl_psgefabk.prj_projekte_id

fk__prj_pl_psgefabk__ps_gefabk_id aus Spalte prj.pl_psgefabk.ps_gefabk_id

fk__prj_pl_psgefkomp__prj_projekte_id aus Spalte prj.pl_psgefkomp.prj_projekte_id

fk__prj_pl_psgefkomp__ps_gefkomp_id aus Spalte prj.pl_psgefkomp.ps_gefkomp_id

fk__prj_pl_psgefnetz__prj_projekte_id aus Spalte prj.pl_psgefnetz.prj_projekte_id

fk__prj_pl_psgefnetz__ps_gefnetz_id aus Spalte prj.pl_psgefnetz.ps_gefnetz_id

fk__prj_pl_psgefprot__prj_projekte_id aus Spalte prj.pl_psgefprot.prj_projekte_id

fk__prj_pl_psgefprot__ps_gefprot_id aus Spalte prj.pl_psgefprot.ps_gefprot_id

fk__prj_pl_psgefstrkt__prj_projekte_id aus Spalte prj.pl_psgefstrkt.prj_projekte_id

fk__prj_pl_psgefstrkt__ps_gefstrkt_id aus Spalte prj.pl_psgefstrkt.ps_gefstrkt_id

fk__prj_plu_psgefanm__prj_projekte_id aus Spalte prj.plu_psgefanm.prj_projekte_id

fk__prj_plu_psgefanm__ps_gefanm_id aus Spalte prj.plu_psgefanm.ps_gefanm_id

fk__prj_plu_psgefanm__spr_uebvar_id aus Spalte prj.plu_psgefanm.spr_uebvar_id

fk__prj_pp_dokausgabe__dok_ausgabe_id aus Spalte prj.pp_dokausgabe.dok_ausgabe_id

fk__prj_pp_dokausgabe__prj_personen_id aus Spalte prj.pp_dokausgabe.prj_personen_id

fk__prj_pp_dokausgabe__prj_projekte_id aus Spalte prj.pp_dokausgabe.prj_projekte_id

fk__prj_ppu_kmpkomp__kmp_komp_id aus Spalte prj.ppu_kmpkomp.kmp_komp_id

fk__prj_ppu_kmpkomp__prj_personen_id aus Spalte prj.ppu_kmpkomp.prj_personen_id

fk__prj_ppu_kmpkomp__prj_projekte_id aus Spalte prj.ppu_kmpkomp.prj_projekte_id

fk__prj_ppu_kmpkomp__spr_uebvar_id__fzg aus Spalte prj.ppu_kmpkomp.spr_uebvar_id__fzg

fk__prj_ppu_prtprotokoll__prj_personen_id aus Spalte prj.ppu_prtprotokoll.prj_personen_id

fk__prj_ppu_prtprotokoll__prj_projekte_id aus Spalte prj.ppu_prtprotokoll.prj_projekte_id

fk__prj_ppu_prtprotokoll__prt_protokoll_id aus Spalte prj.ppu_prtprotokoll.prt_protokoll_id

fk__prj_ppu_prtprotokoll__spr_uebvar_id aus Spalte prj.ppu_prtprotokoll.spr_uebvar_id

fk__prj_ppu_psgefszenario__prj_personen_id aus Spalte prj.ppu_psgefszenario.prj_personen_id

fk__prj_ppu_psgefszenario__prj_projekte_id aus Spalte prj.ppu_psgefszenario.prj_projekte_id

fk__prj_ppu_psgefszenario__ps_gefszenario_id aus Spalte prj.ppu_psgefszenario.ps_gefszenario_id

fk__prj_ppu_psgefszenario__spr_uebvar_id aus Spalte prj.ppu_psgefszenario.spr_uebvar_id

fk__prj_ppu_psgefszenario__spr_uebvar_id__ursp aus Spalte prj.ppu_psgefszenario.spr_uebvar_id__u

fk__prj_ppu_strstrkt__prj_personen_id aus Spalte prj.ppu_strstrkt.prj_personen_id

fk__prj_ppu_strstrkt__prj_projekte_id aus Spalte prj.ppu_strstrkt.prj_projekte_id

fk__prj_ppu_strstrkt__spr_uebvar_id aus Spalte prj.ppu_strstrkt.spr_uebvar_id

fk__prj_ppu_strstrkt__str_strkt_id aus Spalte prj.ppu_strstrkt.str_strkt_id

fk__prj_pu_psabk__prj_projekte_id aus Spalte prj.pu_psabk.prj_projekte_id

fk__prj_pu_psabk__ps_abk_id aus Spalte prj.pu_psabk.ps_abk_id

fk__prj_pu_psabk__spr_uebvar_id aus Spalte prj.pu_psabk.spr_uebvar_id

fk__prj_pu_psabk__spr_uebvar_id__l aus Spalte prj.pu_psabk.spr_uebvar_id__l

fk__prj_pu_strverbindung__prj_projekte_id aus Spalte prj.pu_strverbindung.prj_projekte_id

fk__prj_pu_strverbindung__str_verbindung_id aus Spalte prj.pu_strverbindung.str_verbindung_id

Unique Constraints

uc__prj_p_dokkomp in Tabelle prj.p_dokkomp für die Spalte(n) prj.p_dokkomp.idfrei, prj.p_dokkomp.do
prj.p_dokkomp.prj_projekte_id

uc__prj_p_dokstrkt in Tabelle prj.p_dokstrkt für die Spalte(n) prj.p_dokstrkt.idfrei,
prj.p_dokstrkt.dok_strkt_id, prj.p_dokstrkt.prj_projekte_id

uc_prj_p_kmpstrkt in Tabelle prj.p_kmpstrkt für die Spalte(n) prj.p_kmpstrkt.idfrei, prj.p_kmpstrkt.kmp_strkt_id, prj.p_kmpstrkt.prj_projekte_id

uc_prj_p_prtdokausgabe in Tabelle prj.p_prtdokausgabe für die Spalte(n) prj.p_prtdokausgabe.id prj.p_prtdokausgabe.prt_dokausgabe_id, prj.p_prtdokausgabe.prj_projekte_id

uc_prj_p_prtkomp in Tabelle prj.p_prtkomp für die Spalte(n) prj.p_prtkomp.idfrei, prj.p_prtkom prj.p_prtkomp.prj_projekte_id

uc_prj_p_prtstrkt in Tabelle prj.p_prtstrkt für die Spalte(n) prj.p_prtstrkt.idfrei, prj.p_prtstrkt.prj_projekte_id, prj.p_prtstrkt.prt_strkt_id

uc_prj_p_prtverbindung in Tabelle prj.p_prtverbindung für die Spalte(n) prj.p_prtverbindung.id prj.p_prtverbindung.prj_projekte_id, prj.p_prtverbindung.prt_verbindung_id

uc_prj_personen in Tabelle prj.personen für die Spalte(n) prj.personen.idfrei, prj.personen.n prj.personen.name_v, prj.personen.intern_firma_id

uc_prj_pl_psgefabk in Tabelle prj.pl_psgefabk für die Spalte(n) prj.pl_psgefabk.idfrei, prj.pl_psgefabk.prj_projekte_id, prj.pl_psgefabk.ps_gefabk_id

uc_prj_pl_psgefkomp in Tabelle prj.pl_psgefkomp für die Spalte(n) prj.pl_psgefkomp.idfrei, prj.pl_psgefkomp.prj_projekte_id, prj.pl_psgefkomp.ps_gefkomp_id

uc_prj_pl_psgefnetz in Tabelle prj.pl_psgefnetz für die Spalte(n) prj.pl_psgefnetz.idfrei, prj.pl_psgefnetz.prj_projekte_id, prj.pl_psgefnetz.ps_gefnetz_id

uc_prj_pl_psgefprot in Tabelle prj.pl_psgefprot für die Spalte(n) prj.pl_psgefprot.idfrei, prj.pl_psgefprot.prj_projekte_id, prj.pl_psgefprot.ps_gefprot_id

uc_prj_pl_psgefstrkt in Tabelle prj.pl_psgefstrkt für die Spalte(n) prj.pl_psgefstrkt.idfrei, prj.pl_psgefstrkt.prj_projekte_id, prj.pl_psgefstrkt.ps_gefstrkt_id

uc_prj_plu_psgefanm in Tabelle prj.plu_psgefanm für die Spalte(n) prj.plu_psgefanm.idfrei, prj.plu_psgefanm.prj_projekte_id, prj.plu_psgefanm.ps_gefanm_id

uc_prj_pp_dokausgabe in Tabelle prj.pp_dokausgabe für die Spalte(n) prj.pp_dokausgabe.idfrei, prj.pp_dokausgabe.dok_ausgabe_id, prj.pp_dokausgabe.prj_personen_id, prj.pp_dokausg

uc_prj_ppu_kmpkomp in Tabelle prj.ppu_kmpkomp für die Spalte(n) prj.ppu_kmpkomp.idfrei, prj.ppu_kmpkomp.kmp_komp_id, prj.ppu_kmpkomp.prj_personen_id, prj.ppu_kmpkomp.prj_p

uc_prj_ppu_prtprotokoll in Tabelle prj.ppu_prtprotokoll für die Spalte(n) prj.ppu_prtprotokoll prj.ppu_prtprotokoll.prj_projekte_id, prj.ppu_prtprotokoll.prt_protokoll_id

uc_prj_ppu_psgefszenario in Tabelle prj.ppu_psgefszenario für die Spalte(n) prj.ppu_psgefszena prj.ppu_psgefszenario.prj_personen_id, prj.ppu_psgefszenario.prj_projekte_id, prj.ppu_psgefszenario.ps_gefszenario_id

uc_prj_ppu_strstrkt in Tabelle prj.ppu_strstrkt für die Spalte(n) prj.ppu_strstrkt.idfrei, prj.ppu_strstrkt.prj_personen_id, prj.ppu_strstrkt.prj_projekte_id, prj.ppu_strstrk

uc__prj_projekte in Tabelle prj.projekte für die Spalte(n) prj.projekte.idfrei, prj.projekte.prjna

uc__prj_pu_psabk in Tabelle prj.pu_psabk für die Spalte(n) prj.pu_psabk.idfrei, prj.pu_psabk.prj_pr
prj.pu_psabk.ps_abk_id

uc__prj_pu_strverbindung in Tabelle prj.pu_strverbindung für die Spalte(n) prj.pu_strverbindung.idf
prj.pu_strverbindung.prj_projekte_id, prj.pu_strverbindung.str_verbindung_id

6.10 prj.nimm__tabwerte

Funktionsname
prj.nimm__tabwerte(IN tabname varchar, IN tabwerte varchar[], OUT tabid varchar
)

Funktionsparameter/Rückgabewert(e)

tabname Name der Tabelle (ohne Schemanamen) des Schemas prj, welche die Werte übernehmen soll (Details siehe Funktionsbeschreibung — Werteliste einlesen).

tabwerte Liste der Werte, welche in die aktuelle Tabelle eingelesen werden sollen (Details siehe Funktionsbeschreibung — Werteliste einlesen).

tabid Es wird die Identifikationsnummer des aktuell eingelesenen Wertes prj.*.id zurückgegeben.

Funktionsbeschreibung
Mittels dieser Funktion werden in die Tabellen des Schemas prj die Werte eingelesen.

Werteliste einlesen:
Dem Funktionsparameter tabwerte ist eine Werteliste (ARRAY) zu übergeben (die maximale Anzahl und die Reihenfolge der Werte ist fest vorgegeben — siehe oben). Soll nur eine Teilmenge der Werte übergeben werden, dann sind alle Werte von links nach rechts zu spezifizieren (nichtbenötigte Werte am Ende der Liste dürfen weggelassen werden).

p_dokkomp, p_dokstrkt, p_kmpstrkt, p_prtdokausgabe, p_prtkomp, p_prtstrkt, p_prtverbindung

1. prj_projekte_id
2. dok_komp_id, dok_strkt_id, kmp_strkt_id, prt_dokausgabe_id, prt_komp_id, prt_strkt_id, prt_verbindung_id
3. notizen

Es müssen mindestens die ersten beiden Werte übergeben werden. Sie müssen größer 0 sein.
personen

1. intern_firma_id
2. name_n

3. name_v
4. notizen

Es müssen mindestens die ersten drei Werte übergeben werden. Der erste Wert muß größer 0 sein.

pl_psgefabk, pl_psgefkomp, pl_psgefnetz, pl_psgefprot, pl_psgefstrkt

1. prj_projekte_id
2. ps_gefabk_id, ps_gefkomp_id, ps_gefnetz_id, ps_gefprot_id, ps_gefstrkt_id
3. logdatuhrz
4. logdquelle
5. logname
6. logdqok
7. notizen

Es müssen mindestens die ersten beiden Werte übergeben werden. Sie müssen größer 0 sein.

plu_psgefanm

1. prj_projekte_id
2. ps_gefanm_id
3. spr_uebvar_id
4. logdatuhrz
5. logdquelle
6. logname
7. logdqok
8. notizen

Es müssen mindestens die ersten beiden Werte übergeben werden. Die ersten drei Werte müssen größer 0 sein.

pp_dokausgabe, ppu_kmpkomp, ppu_prtprotokoll, ppu_psgefszenario, ppu_strstrkt

1. prj_projekte_id
2. dok_ausgabe_id, kmp_komp_id, prt_protokoll_id, ps_gefszenario_id, str_strkt_id
3. prj_personen_id
4. notizen, nrfzg, teilnehmer, spr_uebvar_id, spr_uebvar_id Für die Spalten *_id muß der Wert größer 0 sein.

5. `-, spr_uebvar_id__fzg, datuhrz, spr_uebvar_id__ursp, fbereignis` Für die Spalten `*_id_*` muß der Wert größer `0` sein.
6. `-, notizen, duanfrage, notizen, notizen`
7. `-, -, termin`
8. `-, -, intern`
9. `-, -, erledigt`
10. `-, -, spr_uebvar_id` Der Wert muß größer `0` sein.
11. `-, -, notizen`

Bei den Tabellen **pp_dokausgabe, ppu_kmpkomp** müssen die ersten drei und bei den restlichen Tabellen die ersten zwei Werte übergeben werden. Die ersten drei Werte müssen größer `0` sein.
projekte

1. `prjname`
2. `prjbeschr`
3. `makefile`
4. `notizen`

Es müssen mindestens die ersten beiden Werte übergeben werden.
pu_psabk

1. `prj_projekte_id`
2. `ps_abk_id`
3. `spr_uebvar_id`
4. `spr_uebvar_id__l`
5. `notizen`

Es müssen mindestens die ersten beiden Werte übergeben werden.
pu_strverbindung

1. `prj_projekte_id`
2. `str_verbindung_id`
3. `fbtor`
4. `notizen`

Es müssen mindestens die ersten beiden Werte übergeben werden.

Beispiele

```
-- Fehlermeldung. Ungültige Tabelle !
SELECT * FROM prj.nimm__tabwerte( NULL, NULL );

-- Fehlermeldung. Bei allen Tabellen müssen mindestens zwei Elemente übergeben
werden
SELECT * FROM prj.nimm__tabwerte( 'projekte', ARRAY['1'] );

-- Fehlermeldung. Die übergebenen Parameter müssen größer 0 sein.
SELECT * FROM prj.nimm__tabwerte( 'p_dokkomp', ARRAY['0', '1'] );
SELECT * FROM prj.nimm__tabwerte( 'p_dokkomp', ARRAY['1', '0'] );
SELECT * FROM prj.nimm__tabwerte( 'plu_psgefanm', ARRAY['1', '1', '0'] );
SELECT * FROM prj.nimm__tabwerte( 'ppu_kmpkomp', ARRAY['1', '1', '1', '', '0'] );
SELECT * FROM prj.nimm__tabwerte( 'ppu_prtprotokoll', ARRAY['1', '1', '1', '', '',
'', '', '', '', '', '0'] );

-- Fehlermeldung. Es dürfen nur die zulässigen Begriffe verwendet werden
SELECT * FROM prj.nimm__tabwerte( 'ppu_strstrkt', ARRAY['1', '22', '1', '99', 'ttt']
);
SELECT * FROM prj.nimm__tabwerte( 'pu_strverbindung', ARRAY['1', '1', 'ttt'] );

-- prj.projekte einlesen
SELECT * FROM prj.projekte;
SELECT * FROM prj.nimm__tabwerte( 'projekte', ARRAY['testProj', 'Testprojekt',
'Inhalt des makefile', 'Zusatzinformationen'] );
SELECT * FROM prj.projekte;

-- dok.* einlesen
SELECT * FROM prj.pp_dokausgabe;
SELECT * FROM prj.nimm__tabwerte( 'pp_dokausgabe', ARRAY[prj.nimm__tabwerte( 'projekte'
ARRAY['testProj', 'Testprojekt'] ), dok.nimm__tabwerte( 'ausgabe', ARRAY[ dok.nimm__dok
ARRAY['08-15', '1'] ), '01'] ), prj.nimm__tabwerte( 'personen', ARRAY['1', 'Bellair',
'Michael'] ), 'Zusatzinformationen'] );
SELECT * FROM prj.pp_dokausgabe;
SELECT * FROM prj.p_dokkomp;
SELECT * FROM prj.nimm__tabwerte( 'p_dokkomp', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), dok.nimm__tabwerte( 'komp', ARRAY[dok.nimm__doku(
ARRAY['08-15', '1'] ), kmp.nimm__komp( ARRAY['1', '47/11'] )] ), 'Zusatzinformationen'
);
SELECT * FROM prj.p_dokkomp;
SELECT * FROM prj.p_dokstrkt;
SELECT * FROM prj.nimm__tabwerte( 'p_dokstrkt', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), dok.nimm__tabwerte( 'strkt', ARRAY[dok.nimm__doku(
ARRAY['08-15', '1'] ), str.nimm__strkt( ARRAY['1', '-BAAA', spr.nimm__wgruppe( 'Struktu
```

```
'1', '0' )] )] ), 'Zusatzinformationen'] );
SELECT * FROM prj.p_dokstrkt;

-- kmp.* einlesen
SELECT * FROM prj.ppu_kmpkomp;
SELECT * FROM prj.nimm__tabwerte( 'ppu_kmpkomp', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), kmp.nimm__komp( ARRAY['1', '08-15'] ), prj.nimm__tabwer
'personen', ARRAY['1', 'Bellair', 'Michael'] ), 'kmp-id-fzg', spr.nimm__wgruppe(
'Komponentenbezeichnung-Fahrezeughersteller', '1', '0' ), 'Zusatzinformationen']
);
SELECT * FROM prj.ppu_kmpkomp;
SELECT * FROM prj.p_kmpstrkt;
SELECT * FROM prj.nimm__tabwerte( 'p_kmpstrkt', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), kmp.nimm__strkt( ARRAY[kmp.nimm__komp( ARRAY['1',
'08-15'] ), str.nimm__strkt( ARRAY['1', '-BAAA', spr.nimm__wgruppe( 'Strukturpunkt',
'1', '0' )] ), 'Zusatzinformationen'] ), 'Zusatzinformationen'] );
SELECT * FROM prj.p_kmpstrkt;

-- prt.* einlesen
SELECT * FROM prj.p_prtdokausgabe;
SELECT * FROM prj.nimm__tabwerte( 'p_prtdokausgabe', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), prt.nimm__tabwerte( 'dokausgabe', ARRAY[prt.nimm__proto
ARRAY[spr.nimm__wgruppe( 'Aktueller Protokolleintrag.', '1', '0' ), 'PE_OK'] ),
dok.nimm__tabwerte( 'ausgabe', ARRAY[ dok.nimm__doku( ARRAY['08-15', '1'] ), '01']
), 'Zusatzinformationen'] ), 'Zusatzinformationen'] );
SELECT * FROM prj.p_prtdokausgabe;
SELECT * FROM prj.p_prtkomp;
SELECT * FROM prj.nimm__tabwerte( 'p_prtkomp', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), prt.nimm__tabwerte( 'komp', ARRAY[prt.nimm__protokollei
ARRAY[spr.nimm__wgruppe( 'Aktueller Protokolleintrag.', '1', '0' ), 'PE_OK'] ),
kmp.nimm__komp( ARRAY['1', '47/11'] )] ), 'Zusatzinformationen'] );
SELECT * FROM prj.p_prtkomp;
SELECT * FROM prj.ppu_prtprotokoll;
SELECT * FROM prj.nimm__tabwerte( 'ppu_prtprotokoll', ARRAY[prj.nimm__tabwerte(
'projekte', ARRAY['testProj', 'Testprojekt'] ), prt.nimm__protokolleintrag( ARRAY[spr.nimm
'Aktueller Protokolleintrag.', '1', '0' ), 'PE_OK'] ), prj.nimm__tabwerte( 'personen',
ARRAY['1', 'Bellair', 'Michael'] ), 'true', '2007-06-08', '2007-06-20', '2007-07-07',
'true', 'true', spr.nimm__wgruppe( 'Projektabhängiger, aktueller Protokolleintrag.',
'1', '0' ), 'Zusatzinformationen'] );
SELECT * FROM prj.ppu_prtprotokoll;
SELECT * FROM prj.p_prtstrkt;
SELECT * FROM prj.nimm__tabwerte( 'p_prtstrkt', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), prt.nimm__tabwerte( 'strkt', ARRAY[prt.nimm__protokolle
ARRAY[spr.nimm__wgruppe( 'Aktueller Protokolleintrag.', '1', '0' ), 'PE_OK'] ),
str.nimm__strkt( ARRAY['1', '-BAAA', spr.nimm__wgruppe( 'Strukturpunkt', '1', '0'
)])]), 'Zusatzinformationen'] );
```

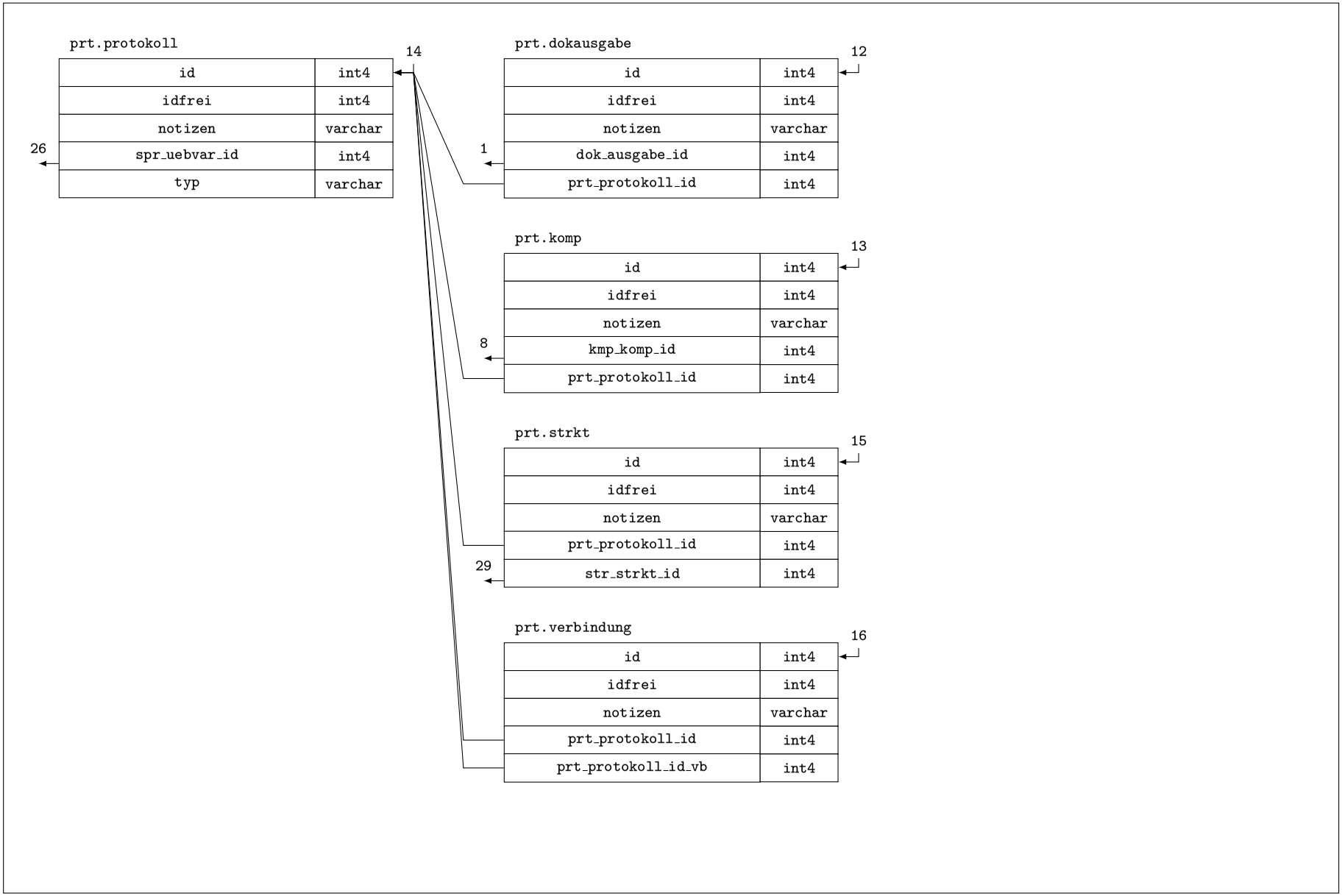
```
SELECT * FROM prj.p_prtstrkt;
SELECT * FROM prj.p_prtverbindung;
SELECT * FROM prj.nimm__tabwerte( 'p_prtverbindung', ARRAY[prj.nimm__tabwerte( 'projekt
ARRAY['testProj', 'Testprojekt'] ), prt.nimm__tabwerte( 'verbindung', ARRAY[prt.nimm__p
ARRAY[spr.nimm__wgruppe( 'Aktueller Protokolleintrag.', '1', '0' ), 'PE_OK'] ),
prt.nimm__protokolleintrag( ARRAY[spr.nimm__wgruppe( 'Zugeordneter Aktueller Protokolle
'1', '0' ), 'PE_OK'] )]), 'Zusatzinformationen'] );
SELECT * FROM prj.p_prtverbindung;

-- ps.* einlesen
SELECT * FROM prj.pu_psabk;
SELECT * FROM prj.nimm__tabwerte( 'pu_psabk', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), ps.nimm__abk( ARRAY[spr.nimm__wgruppe( 'abk',
'1', '0' ), spr.nimm__wgruppe( 'Abkürzung', '1', '0' ), 'BP'] ), spr.nimm__wgruppe(
'projabk', '1', '0' ), spr.nimm__wgruppe( 'projektabhängige Abkürzung', '1', '0'
), 'Zusatzinformationen'] );
SELECT * FROM prj.pu_psabk;
SELECT * FROM prj.pl_psgefabk;
SELECT * FROM prj.nimm__tabwerte( 'pl_psgefabk', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), ps.nimm__aspekte( 'gefabk', ARRAY[ ps.nimm__gefszen
ARRAY[spr.nimm__wgruppe( 'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung
des Gefahrenszenarios', '1', '0' )] ), ps.nimm__abk( ARRAY[spr.nimm__wgruppe( 'abk',
'1', '0' ), spr.nimm__wgruppe( 'Abkürzung', '1', '0' ), 'BP'] )] ), '2007-06-06',
'datenquelle', 'micha', 'true', 'Zusatzinformationen'] );
SELECT * FROM prj.pl_psgefabk;
SELECT * FROM prj.plu_psgefanm;
SELECT * FROM prj.nimm__tabwerte( 'plu_psgefanm', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), ps.nimm__aspekte( 'gefanm', ARRAY[ps.nimm__gefszena
ARRAY[spr.nimm__wgruppe( 'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung
des Gefahrenszenarios', '1', '0' )] ), spr.nimm__wgruppe( 'Anmerkungen', '1', '0'
)] ), spr.nimm__wgruppe( 'projektspezifische Anmerkungen', '1', '0' ), '2007-06-06',
'datenquelle', 'micha', 'true', 'Zusatzinformationen'] );
SELECT * FROM prj.plu_psgefanm;
SELECT * FROM prj.pl_psgefkomp;
SELECT * FROM prj.nimm__tabwerte( 'pl_psgefkomp', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), ps.nimm__aspekte( 'gefkomp', ARRAY[ps.nimm__gefszen
ARRAY[spr.nimm__wgruppe( 'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung
des Gefahrenszenarios', '1', '0' )] ), kmp.nimm__komp( ARRAY['1','08-15'] )] ),
'2007-06-06', 'datenquelle', 'micha', 'true', 'Zusatzinformationen'] );
SELECT * FROM prj.pl_psgefkomp;
SELECT * FROM prj.pl_psgefnetz;
SELECT * FROM prj.nimm__tabwerte( 'pl_psgefnetz', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), ps.nimm__aspekte( 'gefnetz', ARRAY[ps.nimm__gefszen
ARRAY[spr.nimm__wgruppe( 'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung
des Gefahrenszenarios', '1', '0' )]), str.nimm__strkt( ARRAY['1', '-BAAA', spr.nimm__wg
'Strukturpunkt', '1', '0' )])] ), '2007-06-06', 'datenquelle', 'micha', 'true',
```

```
'Zusatzinformationen'] );
SELECT * FROM prj.pl_psgefnetz;
SELECT * FROM prj.pl_psgefprot;
SELECT * FROM prj.nimm__tabwerte( 'pl_psgefprot', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), ps.nimm__aspekte( 'gefprot', ARRAY[ps.nimm__gefszenario
ARRAY[spr.nimm__wgruppe( 'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung
des Gefahrenszenarios', '1', '0' )]), prt.nimm__protokolleintrag( ARRAY[spr.nimm__wgruppe(
'Zugeordneter Aktueller Protokolleintrag.', '1', '0' ), 'PE_OK'] )] ), '2007-06-06',
'datenquelle', 'micha', 'true', 'Zusatzinformationen'] );
SELECT * FROM prj.pl_psgefprot;
SELECT * FROM prj.pl_psgefstrkt;
SELECT * FROM prj.nimm__tabwerte( 'pl_psgefstrkt', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), ps.nimm__aspekte( 'gefstrkt', ARRAY[ps.nimm__gefszenari
ARRAY[spr.nimm__wgruppe( 'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung
des Gefahrenszenarios', '1', '0' )]), str.nimm__strkt( ARRAY['1', '-BAAA', spr.nimm__wgrupp
'Strukturpunkt', '1', '0' )])] ), '2007-06-06', 'datenquelle', 'micha', 'true',
'Zusatzinformationen'] );
SELECT * FROM prj.pl_psgefstrkt;
SELECT * FROM prj.ppu_psgefszenario;
SELECT * FROM prj.nimm__tabwerte( 'ppu_psgefszenario', ARRAY[prj.nimm__tabwerte(
'projekte', ARRAY['testProj', 'Testprojekt'] ), ps.nimm__gefszenario( ARRAY[spr.nimm__wgrup
'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung des Gefahrenszenarios',
'1', '0' )]), prj.nimm__tabwerte( 'personen', ARRAY['1', 'Bellair', 'Michael']
), spr.nimm__wgruppe( 'projektspezifisch formuliertes Gefahrenszenario', '1', '0'
), spr.nimm__wgruppe( 'projektspezifisch formulierter Ursprung des Gefahrenszenarios',
'1', '0' ), 'Zusatzinformationen'] );
SELECT * FROM prj.ppu_psgefszenario;

-- str.* einlesen
SELECT * FROM prj.ppu_strstrkt;
SELECT * FROM prj.nimm__tabwerte( 'ppu_strstrkt', ARRAY[prj.nimm__tabwerte( 'projekte',
ARRAY['testProj', 'Testprojekt'] ), str.nimm__strkt( ARRAY['1', '-BAAA', spr.nimm__wgruppe(
'Strukturpunkt', '1', '0' )]), prj.nimm__tabwerte( 'personen', ARRAY['1', 'Bellair',
'Michael'] ), spr.nimm__wgruppe( 'Strukturpunkt übersetzt ?', '1', '0' ), 'HAUS',
'Zusatzinformationen'] );
SELECT * FROM prj.ppu_strstrkt;
SELECT * FROM prj.pu_strverbindung;
SELECT * FROM prj.nimm__tabwerte( 'pu_strverbindung', ARRAY[prj.nimm__tabwerte(
'projekte', ARRAY['testProj', 'Testprojekt'] ), str.nimm__verbindung( ARRAY[str.nimm__strkt
ARRAY['1', '-BAAA1', spr.nimm__wgruppe( 'Strukturpunkt-1', '1', '0' )]), str.nimm__strkt(
ARRAY['1', '-BAAA2', spr.nimm__wgruppe( 'Strukturpunkt-2', '1', '0' )])] ), 'UND',
'Zusatzinformationen'] );
SELECT * FROM prj.pu_strverbindung;
```

7 prt

Im Schema „Protokoll“ werden neben Protokolleinträgen (PE) auch Sicherheitsanforderungen (SF) und Terminplaneinträge (TP) gespeichert.
Mittels der Tabelle `prt.verbindung` können Verbindungn zwischen den PE, SF, TP hergestellt werden (sind auch aspektübergreifend [PE - SF - TE] sinnvoll).

7.1 Tabellenhierarchie der Ebenen 1 — 3



7.2 Schlüssel und Bedingungen

Primärschlüssel

pk__prt_dokausgabe_id für Spalte prt.dokausgabe.id

pk__prt_komp_id für Spalte prt.komp.id

pk__prt_protokoll_id für Spalte prt.protokoll.id

pk__prt_strkt_id für Spalte prt.strkt.id

pk__prt_verbindung_id für Spalte prt.verbindung.id

Fremdschlüssel

fk__prt_dokausgabe__dok_ausgabe_id aus Spalte prt.dokausgabe.dok_ausgabe_id

fk__prt_dokausgabe__prt_protokoll_id aus Spalte prt.dokausgabe.prt_protokoll_id

fk__prt_komp__kmp_komp_id aus Spalte prt.komp.kmp_komp_id

fk__prt_komp__prt_protokoll_id aus Spalte prt.komp.prt_protokoll_id

fk__prt_protokoll__spr_uebvar_id aus Spalte prt.protokoll.spr_uebvar_id

fk__prt_strkt__prt_protokoll_id aus Spalte prt.strkt.prt_protokoll_id

fk__prt_strkt__str_strkt_id aus Spalte prt.strkt.str_strkt_id

fk__prt_verbindung__prt_protokoll_id aus Spalte prt.verbindung.prt_protokoll_id

fk__prt_verbindung__prt_protokoll_id_vb aus Spalte prt.verbindung.prt_protokoll_id_vb

Unique Constraints

uc__prt_dokausgabe in Tabelle prt.dokausgabe für die Spalte(n) prt.dokausgabe.idfrei, prt.dokausgabe.dok_ausgabe_id, prt.dokausgabe.prt_protokoll_id

uc__prt_komp in Tabelle prt.komp für die Spalte(n) prt.komp.idfrei, prt.komp.kmp_komp_id, prt.komp.prt_protokoll_id

uc__prt_protokoll in Tabelle prt.protokoll für die Spalte(n) prt.protokoll.idfrei, prt.protokoll.spr_uebvar_id, prt.protokoll.typ

uc__prt_strkt in Tabelle prt.strkt für die Spalte(n) prt.strkt.idfrei, prt.strkt.str_strkt_id, prt.strkt.prt_protokoll_id

uc__prt_verbindung in Tabelle prt.verbindung für die Spalte(n) prt.verbindung.idfrei, prt.verbindung.prt_protokoll_id, prt.verbindung.prt_protokoll_id_vb

7.3 prt.nimm__aspekte

Funktionsname
`prt.nimm__aspekte( IN aspekt varchar, IN protokollid varchar, IN liste_id varchar[] )`

Funktionsparameter/Rückgabewert(e)

`aspekt` Einzulesender Aspekt (siehe „Aspekte“).

`protokollid` Identifikationsnummer des Eintrages `prt.protokoll.id`, welchem die Dokumentenreferenz(en), die Komponente(n), der/die Strukturpunkt(e) bzw. die Verbindung(en) zugeordnet werden soll(en).

`liste_id` Liste der Identifikationsnummern der Dokumentenreferenzen `dok.ausgabe.id`, der Komponenten `kmp.komp.id`, Strukturpunkte `str.strkt.id` bzw. Verbindungen `prt.protokoll.id` die dem aktuellen Eintrag zugeordnet werden sollen. Es muß mindestens ein Wert übergeben werden.

Funktionsbeschreibung
Mittels dieser Funktion werden dem aktuellen Eintrag `prt.protokoll.id` die Dokumentenreferenzen, die Komponenten, die Strukturpunkte bzw. die Verbindungen zugeordnet.

Aspekte:
Es ist zu spezifizieren, welcher Aspekt eingelesen werden soll:

dokausgabe Dokumentenausgabe

komp Komponente

strkt Strukturpunkt

verbindung Verbindung zweier Einträge

Beispiele

```
-- Fehlermeldung. Ungültiger Aspekt !
SELECT * FROM prt.nimm__aspekte( NULL, NULL, NULL );

-- Fehlermeldung. Der Parameter protokollid muß größer 0 sein.
SELECT * FROM prt.nimm__aspekte( 'komp', NULL, NULL );
SELECT * FROM prt.nimm__aspekte( 'komp', '', NULL );
SELECT * FROM prt.nimm__aspekte( 'komp', '0', NULL );

-- Fehlermeldung. Die Werteliste darf nicht NULL sein.
SELECT * FROM prt.nimm__aspekte( 'komp', '1', NULL );

-- Fehlermeldung. Leere Werte sind nicht zulässig bzw. 0 ist nicht zulässig
```

```
SELECT * FROM prt.nimm__aspekte( 'strkt', '1', ARRAY[''] );
SELECT * FROM prt.nimm__aspekte( 'strkt', '1', ARRAY['0'] );
SELECT * FROM prt.nimm__aspekte( 'strkt', '1', ARRAY['11', ''] );

-- Komponente einlesen
SELECT prt_protokoll_id, prt_protokoll_id_vb, prt_komp_id, kmp_komp_id, prt_strkt_id,
str_strkt_id, prt_dokausgabe_id, dok_ausgabe_id FROM prt.zeige__protokolleintrag;
SELECT * FROM prt.nimm__protokolleintrag( ARRAY['11', '2007-04-19', 'PE_OK'] );
SELECT * FROM dok.nimm__ausgabe( ARRAY[ dok.nimm__doku( ARRAY['08-15', '1'] ), '01'
] );
SELECT * FROM prt.nimm__aspekte( 'dokausgabe', '1', ARRAY['1'] );
SELECT * FROM prt.nimm__aspekte( 'komp', '1', ARRAY[kmp.nimm__komp( ARRAY['1', '47/11']
)] );
SELECT * FROM prt.nimm__aspekte( 'strkt', '1', ARRAY['11', '22'] );
SELECT * FROM prt.nimm__aspekte( 'verbindung', '1', ARRAY['1'] );
SELECT prt_protokoll_id, prt_protokoll_id_vb, prt_komp_id, kmp_komp_id, prt_strkt_id,
str_strkt_id, prt_dokausgabe_id, dok_ausgabe_id FROM prt.zeige__protokolleintrag;
```

7.4 prt.nimm__protokolleintrag

Funktionsname
`prt.nimm__protokolleintrag( IN tabwerte varchar[], OUT protokollid varchar )`

Funktionsparameter/Rückgabewert(e)

`tabwerte` Liste der Werte, welche in `prt.protokoll` eingelesen werden sollen (Details siehe Funktionsbeschreibung — Werteliste einlesen). Sie werden in der folgenden Reihenfolge in die Spalten

1. `spr_uebvar_id` - Identifikationsnummer der Wortgruppe, welche den Protokolleintrag / die Sicherheitsanforderung / den Terminplaneintrag darstellt. Sie muß größer `0` sein.
2. `typ` (gültige Kurzbezeichnungen siehe Funktionsbeschreibung — Typ - gültige Kurzbezeichnungen. Sie werden auf Plusibilität geprüft.),
3. `notizen`

Es müssen mindestens die ersten beiden Werte übergeben werden.

`protokollid` Es wird die Identifikationsnummer des eingelesenen Protokolleintrages zurückgegeben (`prt.protokoll.id`)

Funktionsbeschreibung
Ziel der Funktion ist das Einlesen eines Protokolleintrages, einer Sicherheitsanforderung, eines Terminplaneintrages.

Werteliste einlesen:
Dem Funktionsparameter `tabwerte` ist eine Werteliste (`ARRAY`) zu übergeben (die maximale Anzahl und die Reihenfolge der Werte ist fest vorgegeben — siehe oben). Soll nur eine Teilmenge der Werte übergeben werden, dann sind alle Werte von links nach rechts zu spezifizieren (nichtbenötigte Werte am Ende der Liste dürfen weggelassen werden).
Es wird empfohlen, zum Einlesen des ersten Parameters die Funktion `spr.nimm__wgruppe(...)` zu verwenden (Hinweise zum Umgang siehe dort). Daher diese Funktion ein (`ARRAY`) zurückgibt, beim Einlesen jedoch nur der erste Wert benötigt wird, ist sie wie folgt aufzurufen:
`varchar[(spr.nimm__wgruppe( 'Protokolleintrag ...', 'de_DE', NULL ))[1]]`

Typ - gültige Kurzbezeichnungen:
Zur Klassifizierung des Eintrages sind die folgenden Kurzbezeichnungen zu verwenden.
Protokolleintrag

PE_B Beratung

PE_F Frage

PE_PG persönliches Gespräch

PE_SP Sicherheitsüberprüfung erforderlich

PE_SPK Sicherheitsüberprüfung mit dem Kunden durchgeführt

PE_SPS Sicherheitsüberprüfung mit dem Spezialisten durchgeführt

PE_T Telefonat

PE_OK der übergeordnete Aspekt ist erledigt (dargestellt mittels des aktuellen Protokolleintrages), es sind keine weiteren Aktivitäten erforderlich.

Sicherheitsanforderung

SF_EP als Entwurfsprinzip

SF_MB als Maßnahme für den Betrieb

SF_MI als Maßnahme für die Instandhaltung

SF_TS als Typ-/Serienprüfung

Terminplaneintrag

TP_MS Meilenstein

TP_ZR Aktivität über einen Zeitraum hinweg

Beispiele

```
-- Fehlermeldung. Die Mindestanzahl der Elemente muß stimmen.
SELECT * FROM prt.nimm__protokolleintrag( ARRAY['1'] );

-- Fehlermeldung. Der Parameter tabwerte[1] muß größer 0 sein.
SELECT * FROM prt.nimm__protokolleintrag( ARRAY['0', 'PE_OK'] );

-- Fehlermeldung. Der Parameter tabwerte[3] muß gültige Werte enthalten.
SELECT * FROM prt.nimm__protokolleintrag( ARRAY['1', 'C'] );

-- Protokolleintrag einlesen
SELECT prt_protokoll_id, spr_uebvar_id, prt_protokoll_typ, prt_protokoll_notizen
FROM prt.zeige__protokolleintrag;
SELECT * FROM prt.nimm__protokolleintrag( ARRAY[spr.nimm__wgruppe( 'Aktueller Protokoll
'1', '0' ), 'PE_OK'] );
SELECT * FROM prt.nimm__protokolleintrag( ARRAY[spr.nimm__wgruppe( 'Aktueller Protokoll
'1', '0' ), 'PE_OK'] );
SELECT * FROM prt.nimm__protokolleintrag( ARRAY[spr.nimm__wgruppe( 'Weiterer Protokolle
'1', '0' ), 'PE_OK', 'interne Notizen zum PE'] );
SELECT prt_protokoll_id, spr_uebvar_id, prt_protokoll_typ, prt_protokoll_notizen
FROM prt.zeige__protokolleintrag;
```

7.5 prt.nimm__tabwerte

Funktionsname

```
prt.nimm__tabwerte( IN tabname varchar, IN tabwerte varchar[], OUT tabid varchar
)
```

Funktionsparameter/Rückgabewert(e)

`tabname` Name der Tabelle (ohne Schemanamen) des Schemas `prt`, welche die Werte übernehmen soll (Details siehe Funktionsbeschreibung — Werteliste einlesen).

`tabwerte` Liste der Werte, welche in die aktuelle Tabelle eingelesen werden sollen (Details siehe Funktionsbeschreibung — Werteliste einlesen).

`tabid` Es wird die Identifikationsnummer des aktuell eingelesenen Wertes `prt.*.id` zurückgegeben.

Funktionsbeschreibung

Mittels dieser Funktion werden in die Tabellen des Schemas `prt` die Werte eingelesen.

Werteliste einlesen:

Dem Funktionsparameter `tabwerte` ist eine Werteliste (`ARRAY`) zu übergeben (die maximale Anzahl und die Reihenfolge der Werte ist fest vorgegeben — siehe unten). Soll nur eine Teilmenge der Werte übergeben werden, dann sind alle Werte von links nach rechts zu spezifizieren (nichtbenötigte Werte am Ende der Liste dürfen weggelassen werden).

dokausgabe, komp, strkt, verbindung dokausgabe

1. `prt_protokoll_id`
2. `dok_ausgabe_id, kmp_komp_id, str_strkt_id, prt_protokoll_id_vb`
3. `notizen`

Es müssen mindestens die ersten beiden Werte übergeben werden. Sie müssen größer `0` sein.

Beispiele

```
-- Fehlermeldung. Ungültige Tabelle !
SELECT * FROM prt.nimm__tabwerte( NULL, NULL );

-- Fehlermeldung. Bei allen Tabellen müssen mindestens zwei Elemente übergeben
werden
SELECT * FROM prt.nimm__tabwerte( 'dokausgabe', ARRAY['1'] );

-- Fehlermeldung. Die übergebenen Parameter müssen größer 0 sein.
SELECT * FROM prt.nimm__tabwerte( 'dokausgabe', ARRAY['0', '1'] );
SELECT * FROM prt.nimm__tabwerte( 'dokausgabe', ARRAY['1', '0'] );

-- Revisionsstand zum Dokument einlesen
SELECT * FROM prt.dokausgabe;
SELECT * FROM prt.nimm__tabwerte( 'dokausgabe', ARRAY[prt.nimm__protokolleintrag(
ARRAY[spr.nimm__wgruppe( 'Aktueller Protokolleintrag.', '1', '0' ), 'PE_OK'] ),
dok.nimm__tabwerte( 'ausgabe', ARRAY[dok.nimm__doku( ARRAY['08-15', '1'] ), '01']
), 'Zustazinformationen'] );
SELECT * FROM prt.dokausgabe;

-- Komponente einlesen
SELECT * FROM prt.komp;
SELECT * FROM prt.nimm__tabwerte( 'komp', ARRAY[prt.nimm__protokolleintrag( ARRAY[spr.nimm_
'Aktueller Protokolleintrag.', '1', '0' ), 'PE_OK'] ), kmp.nimm__komp( ARRAY['1',
'47/11'] ), 'Zustazinformationen'] );
SELECT * FROM prt.komp;

-- Strukturpunkt einlesen
SELECT * FROM prt.strkt;
SELECT * FROM prt.nimm__tabwerte( 'strkt', ARRAY[prt.nimm__protokolleintrag( ARRAY[spr.nimm
'Aktueller Protokolleintrag.', '1', '0' ), 'PE_OK'] ), str.nimm__strkt( ARRAY['1',
'-BAAA', spr.nimm__wgruppe( 'Strukturpunkt', '1', '0' )]), 'Zusatzinformationen']
```

```
);
SELECT * FROM prt.strkt;

-- Verbindung einlesen
SELECT * FROM prt.verbindung;
SELECT * FROM prt.nimm__tabwerte( 'verbindung', ARRAY[prt.nimm__protokolleintrag(
ARRAY[spr.nimm__wgruppe( 'Aktueller Protokolleintrag.', '1', '0' ), 'PE_OK'] ),
prt.nimm__protokolleintrag( ARRAY[spr.nimm__wgruppe( 'Zugeordneter Protokolleintrag.',
'1', '0' ), 'PE_OK'] ), 'Zustazinformationen'] );
SELECT * FROM prt.verbindung;
```

7.6 prt.zeige__protokolleintrag

Beschreibung der Spalten

prt_protokoll_id siehe prt.protokoll.id

spr_uebvar_id siehe prt.protokoll.spr_uebvar_id

prt_protokoll_typ siehe prt.protokoll.typ

prt_protokoll_typ siehe prt.protokoll.notizen

prt_protokoll_id_vb siehe prt.verbindung.prt_protokoll_id_vb

prt_verbindung_notizen siehe prt.verbindung.notizen

prt_komp_id siehe prt.komp.id

kmp_komp_id siehe prt.komp.kmp_komp_id

prt_komp_notizen siehe prt.komp.notizen

prt_strkt_id siehe prt.strkt.id

str_strkt_id siehe prt.strkt.str_strkt_id

prt_strkt_notizen siehe prt.strkt.notizen

prt_dokausgabe_id siehe prt.dokausgabe.id

dok_ausgabe_id siehe prt.dokausgabe.dok_ausgabe_id

prt_dokausgabe_notizen siehe prt.dokausgabe.notizen

Beschreibung der Sicht

Es werden die Protokolleinträge (PE) zurückgegeben.

8 ps

Das Schema „Produktsicherheit“ dient der Verwaltung sämtlicher Aspekte zur Produktsicherheit.

Die Zuordnung von technischen Systemen / Komponenten bzw. Strukturpunkten erfolgt
explizit

- `str.strkt --- kmp.strkt --- kmp.komp --- ps.gefkomp --- ps.gefszenario`
- `str.strkt --- ps.gefstrkt --- ps.gefszenario`

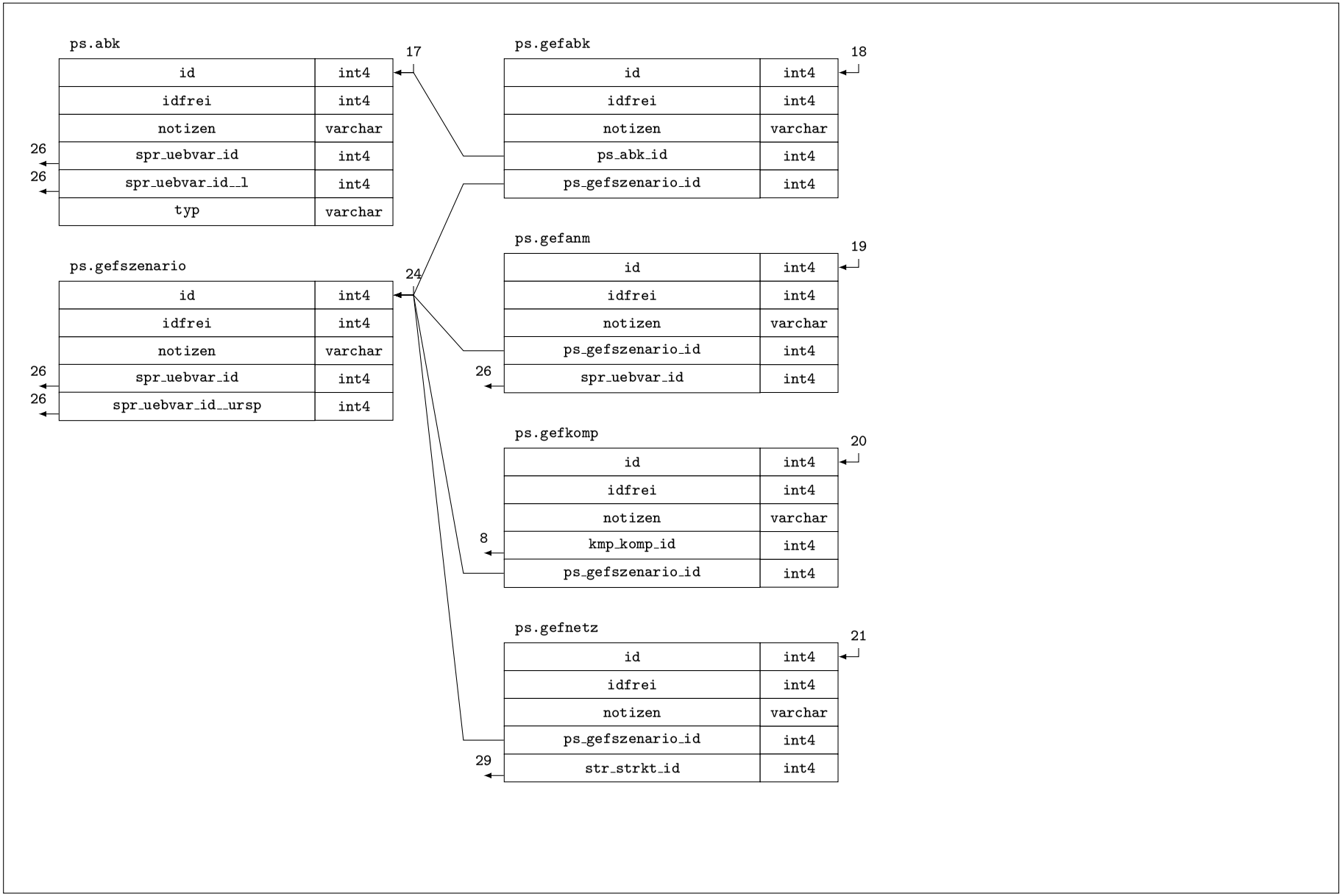
implizit über `prt`

- `str.strkt --- kmp.strkt --- kmp.komp --- prt.komp --- prt.protokoll --- ps.gefprot --- ps.gefszenario`
- `str.strkt --- prt.strkt --- prt.protokoll --- ps.gefprot --- ps.gefszenario`

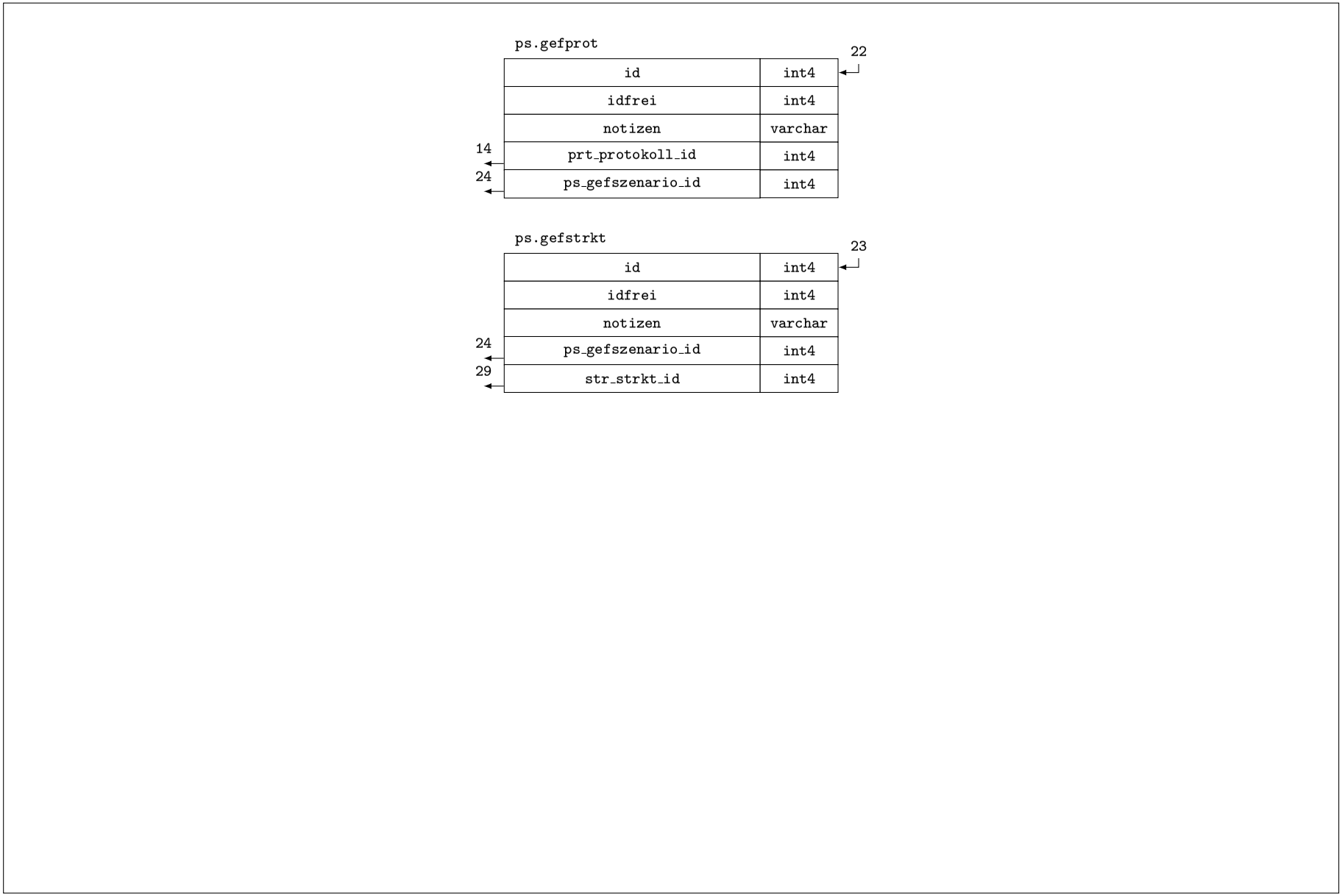
implizit über `prt.dokausgabe`

- `str.strkt --- kmp.strkt --- kmp.komp --- dok.komp --- dok.doku --- dok.ausgabe --- prt.dokausgabe --- prt.protokoll --- ps.gefprot --- ps.gefszenario`
- `str.strkt --- dok.strkt --- dok.doku --- dok.ausgabe --- prt.dokausgabe --- prt.protokoll --- ps.gefprot --- ps.gefszenario`

8.1 Tabellenhierarchie der Ebenen 1 — 3



8.2 Tabellenhierarchie der Ebenen 1 — 3



8.3 Schlüssel und Bedingungen

Primärschlüssel

pk__ps_abk_id für Spalte ps.abk.id

pk__ps_gefabk_id für Spalte ps.gefabk.id

pk__ps_gefanm_id für Spalte ps.gefanm.id

pk__ps_gefkomp_id für Spalte ps.gefkomp.id

pk__ps_gefnetz_id für Spalte ps.gefnetz.id

pk__ps_gefprot_id für Spalte ps.gefprot.id

pk__ps_gefstrkt_id für Spalte ps.gefstrkt.id

pk__ps_gefszenario_id für Spalte ps.gefszenario.id

Fremdschlüssel

fk__ps_abk__spr_uebvar_id aus Spalte ps.abk.spr_uebvar_id

fk__ps_abk__spr_uebvar_id__l aus Spalte ps.abk.spr_uebvar_id__l

fk__ps_gefabk__ps_abk_id aus Spalte ps.gefabk.ps_abk_id

fk__ps_gefabk__ps_gefszenario_id aus Spalte ps.gefabk.ps_gefszenario_id

fk__ps_gefanm__ps_gefszenario_id aus Spalte ps.gefanm.ps_gefszenario_id

fk__ps_gefanm__spr_uebvar_id aus Spalte ps.gefanm.spr_uebvar_id

fk__ps_gefkomp__kmp_komp_id aus Spalte ps.gefkomp.kmp_komp_id

fk__ps_gefkomp__ps_gefszenario_id aus Spalte ps.gefkomp.ps_gefszenario_id

fk__ps_gefnetz__ps_gefszenario_id aus Spalte ps.gefnetz.ps_gefszenario_id

fk__ps_gefnetz__str_strkt_id aus Spalte ps.gefnetz.str_strkt_id

fk__ps_gefprot__prt_protokoll_id aus Spalte ps.gefprot.prt_protokoll_id

fk__ps_gefprot__ps_gefszenario_id aus Spalte ps.gefprot.ps_gefszenario_id

fk__ps_gefstrkt__ps_gefszenario_id aus Spalte ps.gefstrkt.ps_gefszenario_id

fk__ps_gefstrkt__str_strkt_id aus Spalte ps.gefstrkt.str_strkt_id

fk__ps_gefszenario__spr_uebvar_id aus Spalte ps.gefszenario.spr_uebvar_id

fk__ps_gefszenario__spr_uebvar_id__ursp aus Spalte ps.gefszenario.spr_uebvar_id__ursp

Unique Constraints

uc__ps_abk in Tabelle ps.abk für die Spalte(n) ps.abk.idfrei, ps.abk.spr_uebvar_id, ps.abk.typ

uc__ps_gefabk in Tabelle ps.gefabk für die Spalte(n) ps.gefabk.idfrei, ps.gefabk.ps_gefszenario_id, ps.gefabk.ps_abk_id

uc__ps_gefanm in Tabelle ps.gefanm für die Spalte(n) ps.gefanm.idfrei, ps.gefanm.ps_gefszenario_id, ps.gefanm.spr_uebvar_id

uc__ps_gefkomp in Tabelle ps.gefkomp für die Spalte(n) ps.gefkomp.idfrei, ps.gefkomp.ps_gefszenari ps.gefkomp.kmp_komp_id

uc__ps_gefnetz in Tabelle ps.gefnetz für die Spalte(n) ps.gefnetz.idfrei, ps.gefnetz.ps_gefszenari ps.gefnetz.str_strkt_id

uc__ps_gefprot in Tabelle ps.gefprot für die Spalte(n) ps.gefprot.idfrei, ps.gefprot.ps_gefszenari ps.gefprot.prt_protokoll_id

uc__ps_gefstrkt in Tabelle ps.gefstrkt für die Spalte(n) ps.gefstrkt.idfrei, ps.gefstrkt.ps_gefsze ps.gefstrkt.str_strkt_id

uc__ps_gefszenario in Tabelle ps.gefszenario für die Spalte(n) ps.gefszenario.idfrei, ps.gefszenario.spr_uebvar_id, ps.gefszenario.spr_uebvar_id__ursp

8.4 ps.nimm__abk

Funktionsname

```
ps.nimm__abk( IN tabwerte varchar[], OUT abkid varchar )
```

Funktionsparameter/Rückgabewert(e)

tabwerte Liste der Werte, welche in ps.abk eingelesen werden sollen (Details siehe Funktionsbeschreibung — Werteliste einlesen). Sie werden in der folgenden Reihenfolge in die Spalten

1. spr_uebvar_id muß größer 0 sein.
2. spr_uebvar_id__l muß größer 0 sein.
3. typ gültige Kurzbezeichnungen siehe Funktionsbeschreibung — Typ - gültige Kurzbezeichnungen. Sie werden auf Plusibilität geprüft.),
4. notizen

eingetragen. Es müssen mindestens die ersten drei Werte übergeben werden.

abkid Es wird die Identifikationsnummer der Abkürzung ps.abk.id zurückgegeben.

Funktionsbeschreibung

Ziel der Funktion ist das Einlesen der Abkürzungen, welche einem Gefahrenszenario zugeordnet werden können..

Werteliste einlesen:
Dem Funktionsparameter `tabwerte` ist eine Werteliste (`ARRAY`) zu übergeben (die maximale Anzahl und die Reihenfolge der Werte ist fest vorgegeben — siehe oben). Soll nur eine Teilmenge der Werte übergeben werden, dann sind alle Werte von links nach rechts zu spezifizieren (nichtbenötigte Werte am Ende der Liste dürfen weggelassen werden).

Es wird empfohlen, zum Einlesen der ersten beiden Parameter die Funktion `spr.nimm__wgruppe(...)` zu verwenden (Hinweise zum Umgang siehe dort). Daher diese Funktion (`ARRAY`) zurückgibt, beim Einlesen jedoch nur der erste Wert benötigt wird, ist sie wie folgt aufzurufen:
`varchar[(spr.nimm__wgruppe( 'Gefahrenszenario oder Ursprung', 'de_DE', NULL ))[1]]`

Typ - gültige Kurzbezeichnungen:

BP betroffene Person (die dem Gefahrenszenario ausgesetzen Personen, gefährdete Personen)

RA Auswirkungen bei Eintreten des Gefahrenszenarios (zur Risikobewertung lt. EN 50126)

RH Häufigkeit des Auftretens des Gefahrenszenarios (zur Risikobewertung lt. EN 50126)

Beispiele

```
-- Fehlermeldung. Die Mindestanzahl der Elemente muß stimmen.
SELECT * FROM ps.nimm__abk( ARRAY['1', '1'] );

-- Fehlermeldung. Der Parameter tabwerte[1] bzw. tabwerte[2] muß größer 0 sein.
SELECT * FROM ps.nimm__abk( ARRAY['0', '1', 'C' ] );
SELECT * FROM ps.nimm__abk( ARRAY['1', '0', 'C' ] );

-- Fehlermeldung. Der Parameter tabwerte[3] muß gültige Werte enthalten.
SELECT * FROM ps.nimm__abk( ARRAY['1', '1', 'C' ] );

-- Standardverhalten
SELECT * FROM ps.abk;
SELECT * FROM ps.nimm__abk( ARRAY[spr.nimm__wgruppe( 'abk', '1', '0' ), spr.nimm__wgrup
'Abkürzung', '1', '0' ), 'BP'] );
SELECT * FROM ps.nimm__abk( ARRAY[spr.nimm__wgruppe( 'abk-1', '1', '0' ), spr.nimm__wgr
'Abkürzung-1', '1', '0' ), 'BP', 'Zusatzinformationen'] );
SELECT * FROM ps.abk;
```

8.5 ps.nimm__aspekte

Funktionsname

```
ps.nimm__aspekte( IN aspekt varchar, IN tabwerte varchar[], OUT aspektid varchar
```

)

Funktionsparameter/Rückgabewert(e)

`aspekt` Einzulesender Aspekt (siehe „Aspekte“).

`tabwerte` Liste der Werte, welche in die Tabellen der verschiedenen Aspekte eingelesen werden sollen (Details siehe Funktionsbeschreibung — Werteliste einlesen). Sie werden in der folgenden Reihenfolge in die Spalten

1. `ps_gefszenario_id` muß größer `0` sein.
2. `ps_gefabk_id ODER spr_uebvar_id ODER kmp_komp_id ODER str_strkt_id ODER prt_protokoll_id` muß größer `0` sein.
3. `notizen`

eingetragen. Es müssen mindestens die ersten beiden Werte übergeben werden.

`aspektid` Es wird die Identifikationsnummer des aktuelle eingelesenen Aspektes `ps.gef*.id` zurückgegeben.

Funktionsbeschreibung
Mittels dieser Funktion werden dem aktuellen Gefahrenszenario `ps.gefahrenszenario.id` die verschiedenen Aspekte zur Produktsicherheit zugeordnet.

Aspekte:
Es ist zu spezifizieren, welcher Aspekt eingelesen werden soll:

gefabk Aspekte, welche mittels Abkürzung dargestellt werden (z.B. betroffene Personen, Häufigkeit des Auftretens, Schwere der Auswirkung)

gefanm Anmerkungen / Annahmen zu den Maßnahmen zur Risikominderung

gefkomp Komponente, welche in das Gefahrenszenario involviert ist

gefnetz Einordnung des Gefahrenszenarios in das Gefahrennetz

gefprot Gefahrenprotokolleinträge, Sicherheitsanforderungen (implizit Entwurfsprinzipien, organisatorische Maßnahmen), Terminplaneinträge welche dem Gefahrenszenario zuzuordnen sind

gefstrkt Strukturpunkt, welcher dem Gefahrenszenario zuzuordnen ist

Beispiele

```
-- Fehlermeldung. Ungültiger Aspekt !
SELECT * FROM ps.nimm__aspekte( NULL, NULL );

-- Fehlermeldung. Die Mindestanzahl der Elemente muß stimmen.
SELECT * FROM ps.nimm__aspekte( 'gefabk', ARRAY['1'] );
```

```
-- Fehlermeldung. Der Parameter tabwerte[1] bzw. tabwerte[2] muß größer 0 sein.
SELECT * FROM ps.nimm__aspekte( 'gefabk', ARRAY['0', '1'] );
SELECT * FROM ps.nimm__aspekte( 'gefabk', ARRAY['1', '0'] );

-- Abkürzung einlesen
SELECT * FROM ps.gefabk;
SELECT * FROM ps.nimm__aspekte( 'gefabk', ARRAY[ps.nimm__gefszenario( ARRAY[spr.nimm__w
'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung des Gefahrenszenarios',
'1', '0' )]), ps.nimm__abk( ARRAY[spr.nimm__wgruppe( 'abk', '1', '0' ), spr.nimm__wgrup
'Abkürzung', '1', '0' ), 'BP'] ), 'Zusatzinformationen'] );
SELECT * FROM ps.gefabk;

-- Anmerkung einlesen
SELECT * FROM ps.gefanm;
SELECT * FROM ps.nimm__aspekte( 'gefanm', ARRAY[ps.nimm__gefszenario( ARRAY[spr.nimm__w
'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung des Gefahrenszenarios',
'1', '0' )]), spr.nimm__wgruppe( 'Anmerkungen zum Gefahrenszenario', '1', '0' ),
'Zusatzinformationen'] );
SELECT * FROM ps.gefanm;

-- Komponente einlesen
SELECT * FROM ps.gefkomp;
SELECT * FROM ps.nimm__aspekte( 'gefkomp', ARRAY[ps.nimm__gefszenario( ARRAY[spr.nimm__
'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung des Gefahrenszenarios',
'1', '0' )]), kmp.nimm__komp( ARRAY['1', '08-15'] ), 'Zusatzinformationen'] );
SELECT * FROM ps.gefkomp;

-- Gefahrennetz einlesen
SELECT * FROM ps.gefnetz;
SELECT * FROM ps.nimm__aspekte( 'gefnetz', ARRAY[ps.nimm__gefszenario( ARRAY[spr.nimm__
'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung des Gefahrenszenarios',
'1', '0' )]), str.nimm__strkt( ARRAY['1', '-BAAA', spr.nimm__wgruppe( 'Strukturpunkt',
'1', '0' )]), 'Zusatzinformationen'] );
SELECT * FROM ps.gefnetz;

-- Protokolleintrag / Sicherheitsanforderung / Terminplaneintrag einlesen
SELECT * FROM ps.gefprot;
SELECT * FROM ps.nimm__aspekte( 'gefprot', ARRAY[ps.nimm__gefszenario( ARRAY[spr.nimm__
'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung des Gefahrenszenarios',
'1', '0' )]), prt.nimm__protokolleintrag( ARRAY[spr.nimm__wgruppe( 'Aktueller Protokoll
'1', '0' ), 'PE_OK'] ), 'Zusatzinformationen'] );
SELECT * FROM ps.gefprot;

-- Strukturpunkt einlesen
SELECT * FROM ps.gefstrkt;
```

```
SELECT * FROM ps.nimm__aspekte( 'gefstrkt', ARRAY[ps.nimm__gefszenario( ARRAY[spr.nimm__wgru
'Gefahrenszenario', '1', '0' ), spr.nimm__wgruppe( 'Ursprung des Gefahrenszenarios',
'1', '0' )]), str.nimm__strkt( ARRAY['1', '-BAAA', spr.nimm__wgruppe( 'Strukturpunkt',
'1', '0' )]), 'Zusatzinformationen'] );
SELECT * FROM ps.gefstrkt;
```

8.6 ps.nimm__gefszenario

Funktionsname
`ps.nimm__gefszenario( IN tabwerte varchar[], OUT gefszenarioid varchar )`

Funktionsparameter/Rückgabewert(e)

`tabwerte` Liste der Werte, welche in `ps.gefszenario` eingelesen werden sollen (Details siehe Funktionsbeschreibung — Werteliste einlesen). Sie werden in der folgenden Reihenfolge in die Spalten

1. `spr_uebvar_id` - Identifikationsnummer der Wortgruppe, welche das Gefahrenszenario beschreibt. Sie muß größer `0` sein.
2. `spr_uebvar_id__ursp` - Identifikationsnummer der Wortgruppe, welche den Ursprung das Gefahrenszenario beschreibt. Sie muß größer `0` sein.
3. `notizen`

eingetragen. Es müssen mindestens die ersten beiden Werte übergeben werden.

`gefszenarioid` Es wird die Gefahren-Identifikationsnummer `ps.gefszenario.id` zurückgegeben.

Funktionsbeschreibung
Ziel der Funktion ist das Einlesen der Beschreibung und des Ursprungs eines Gefahrenszenarios. Sollte ein Gefahrenszenario mit den spezifizierten Werten bereits im System vorhanden sein, so wird dessen Identifikationsnummer zurückgegeben.

Werteliste einlesen:
Dem Funktionsparameter `tabwerte` ist eine Werteliste (`ARRAY`) zu übergeben (die maximale Anzahl und die Reihenfolge der Werte ist fest vorgegeben — siehe oben). Soll nur eine Teilmenge der Werte übergeben werden, dann sind alle Werte von links nach rechts zu spezifizieren (nichtbenötigte Werte am Ende der Liste dürfen weggelassen werden).

Es wird empfohlen, zum Einlesen der ersten beiden Parameter die Funktion `spr.nimm__wgruppe(...)` zu verwenden (Hinweise zum Umgang siehe dort). Daher diese Funktion (`ARRAY`) zurückgibt, beim Einlesen jedoch nur der erste Wert benötigt wird, ist sie wie folgt aufzurufen:
`varchar[(spr.nimm__wgruppe( 'Gefahrenszenario oder Ursprung', 'de_DE', NULL ))[1]]`

Beispiele

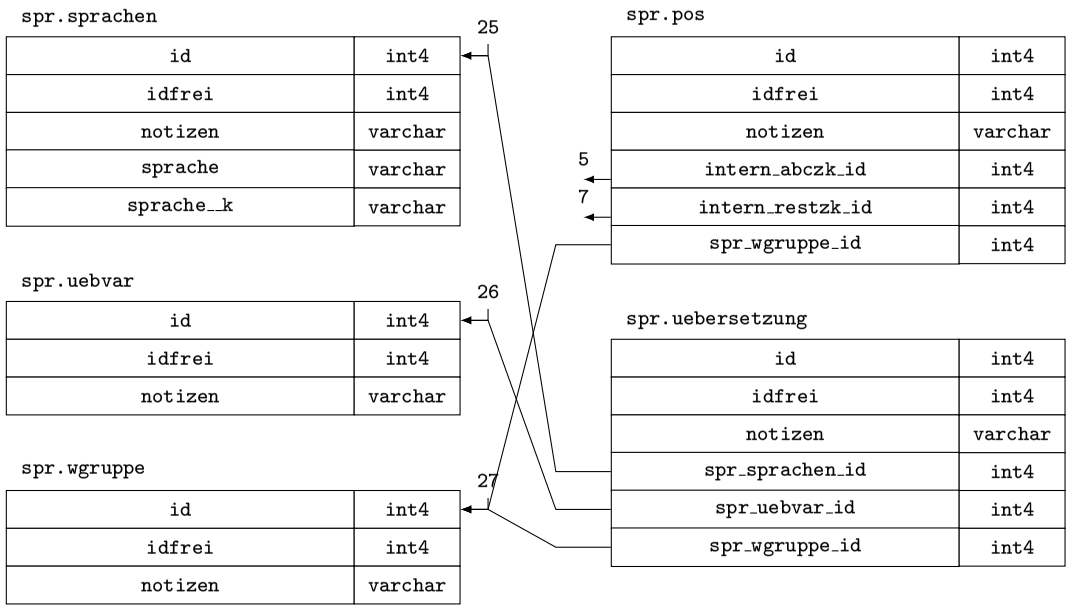
```
Fehlermeldung: Beiden Parametern ist ein Wert größer 0 zu übergeben.
SELECT * FROM ps.nimm__gefszenario( NULL );
SELECT * FROM ps.nimm__gefszenario( ARRAY['-1', '-1'] );
SELECT * FROM ps.nimm__gefszenario( ARRAY['1', '-1'] );

Fehlermeldung: Es sind mindestens zwei Elemente zu üergeben.
SELECT * FROM ps.nimm__gefszenario( ARRAY['-1'] );

PostgreSQL - Fehlermeldung: Verletzte Fremdschlüsselbeziehung !
SELECT * FROM ps.nimm__gefszenario( ARRAY['9999', '1'] );

Standardverhalten
SELECT * FROM ps.gefszenario;
SELECT * FROM ps.nimm__gefszenario( ARRAY[spr.nimm__wgruppe( 'Gefahrenszenario',
'1', '0' ), spr.nimm__wgruppe( 'Ursprung des Gefahrenszenarios', '1', '0' ), 'Zusatzin
SELECT * FROM ps.gefszenario;
```

9 spr

Das Schema `Sprache` dient der sprachenabhängigen Verwaltung von Wortgruppen. Jeder Wortgruppe `spr.wgruppe.id¡` können mehrere unterschiedliche Sprachen `spr.sprachen.id` mittels unterschiedlicher Übersetzungsvarianten `spr.uebvar.id` zugeordnet werden. Die Tabelle `spr.uebersetzung` dient der Zuordnung der Sprachen / Übersetzungsvarianten zur Wortgruppe.
Innerhalb einer Übersetzungsvariante `spr.uebersetzung.spr_uebvar_id` tritt jede Sprache maximal einmal auf.

Die Tabelle `spr.pos` dient der Bildung der Wortgruppen.

9.1 Tabellenhierarchie der Ebenen 1 — 3



9.2 Schlüssel und Bedingungen

Primärschlüssel

pk__spr_pos_id für Spalte spr.pos.id

pk__spr_sprachen_id für Spalte spr.sprachen.id

pk__spr_uebersetzung_id für Spalte spr.uebersetzung.id

pk__spr_uebvar_id für Spalte spr.uebvar.id

pk__spr_wgruppe_id für Spalte spr.wgruppe.id

Fremdschlüssel

fk__spr_pos_intern_abczk_id aus Spalte spr.pos.intern_abczk_id

fk__spr_pos_intern_restzk_id aus Spalte spr.pos.intern_restzk_id

fk__spr_pos_spr_wgruppe_id aus Spalte spr.pos.spr_wgruppe_id

fk__spr_uebersetzung_spr_sprachen_id aus Spalte spr.uebersetzung.spr_sprachen_id

fk__spr_uebersetzung_spr_uebvar_id aus Spalte spr.uebersetzung.spr_uebvar_id

fk__spr_uebersetzung_spr_wgruppe_id aus Spalte spr.uebersetzung.spr_wgruppe_id

Unique Constraints

uc__spr_sprachen in Tabelle spr.sprachen für die Spalte(n) spr.sprachen.idfrei, spr.sprachen.sprac

uc__spr_uebersetzung in Tabelle spr.uebersetzung für die Spalte(n) spr.uebersetzung.idfrei, spr.uebersetzung.spr_sprachen_id, spr.uebersetzung.spr_uebvar_id

Check Constraints

cc__spr_pos in Tabelle spr.pos die Bedingung (((((intern_abczk_id = 0) AND (intern_restzk_id = 0)) AND (spr_wgruppe_id = 0)) OR (((intern_abczk_id > 0) AND (intern_restzk_id = 0)) AND (spr_wgruppe_id > 0))) OR (((intern_abczk_id = 0) AND (intern_restzk_id > 0)) AND (spr_wgruppe_id > 0)))

9.3 spr.nimm__wgruppe

Funktionsname

spr.nimm__wgruppe(IN wortgruppe varchar, IN sprachenid varchar, INOUT spruebvarid varchar)

Funktionsparameter/Rückgabewert(e)

wortgruppe Es ist die einzulesende Wortgruppe anzugeben. Sie darf nicht leer oder NULL sein.

`sprachenid` Es ist die Identifikationsnummer `spr.sprachen.id` der Sprache anzugeben, in der die Wortgruppe eingelesen werden soll. Der Wert muß angegeben werden und größer `0` sein.

`spruebvarid`

Funktionsbeschreibung
Ziel der Funktion ist

- das Einlesen von Wortgruppen im Modus „Einlesen“ bzw.
- das Übersetzen einer im System vorhandenen Referenzwortgruppe `spruebvarid` mittels aktueller Wortgruppe `wortgruppe` in der aktuellen Sprache `spr.sprachen.id` im Modus „Übersetzen“.

Modus „Einlesen“
Konnte die aktuelle Wortgruppe `wortgruppe` zur gewünschten Sprache `spr.sprachen.id` im System gefunden werden, wird die erste passende Übersetzungsvariante `spr.uebvar.id` zurückgegeben. Andernfalls wird sie mit der nächsten verfügbaren `spr.uebvar.id` eingelesen.

Modus „Übersetzen“
Im System werden Übersetzungsvarianten mittels `spr.uebvar.id` gebildet. Sie stellen den konstanten Bezug auf Wortgruppen verschiedener Sprachen, welche in einem gemeinsamen Kontext stehen, dar. Innerhalb einer Übersetzungsvariante darf nur eine Übersetzung je Sprache zugeordnet werden. Somit ist es möglich unter Angabe der Sprache und Identifikationsnummer der Übersetzungsvariante die korrekte Wortgruppe zu ermitteln.
Zum korrekten Verständnis der Funktionalität im Modus „Übersetzen“ sind die folgenden Randbedingungen zu beachten:

1. Wird die aktuelle Wortgruppe zur gewünschten Sprache innerhalb der vorgegebenen Übersetzungsvariante gefunden, so wird die Identifikationsnummer der Übersetzungsvariante zurückgegeben.
2. Wird innerhalb der vorgegebenen Übersetzungsvariante keine Übersetzung in der gewünschten Sprache gefunden, wird die aktuelle Wortgruppe zur gewünschten Sprache eingelesen.
3. Wird die vorgegebenen Übersetzungsvariante nicht gefunden, so wird eine Fehlermeldung ausgegeben und das Einlesen abgebrochen.
4. Wird innerhalb der vorgegebenen Übersetzungsvariante eine Übersetzung in der gewünschten Sprache gefunden (Sprache ist bereits belegt), so wird eine Fehlermeldung ausgegeben und das Einlesen abgebrochen.

Beispiele

```
-- Fehlermeldung, die Wortgruppe ist anzugeben
SELECT * FROM spr.nimm__wgruppe( NULL, '1', NULL );
SELECT * FROM spr.nimm__wgruppe( '', '1', NULL );
```

```
-- Fehlermeldung, die Identifikationsnummer der Sprache muß größer 0 sein
SELECT * FROM spr.nimm__wgruppe( 'Das ist ein Test .', '0', NULL );

-- Fehlermeldung, die Identifikationsnummer der Übersetzungsvariante muß größer/gleich
0 sein
SELECT * FROM spr.nimm__wgruppe( 'Das ist ein Test .', '1', NULL );
SELECT * FROM spr.nimm__wgruppe( 'das ist ein Array', '1', '-1' );

-- Fehlermeldung, die Identifikationsnummer der Übersetzungsvariante muß im System
vorhanden sein
SELECT * FROM spr.nimm__wgruppe( 'das ist ein Array', '1', '999999' );

-- Fehlermeldung, die Übersetzungsvariante ist bereits mit einer Übersetzung in
der gewünschten Sprache belegt
SELECT * FROM spr.nimm__wgruppe( 'Das ist ein Array.', '1', '0' );
SELECT * FROM spr.nimm__wgruppe( 'This is an Array.', '2', '1' );
SELECT * FROM spr.zeige__wgruppe;
SELECT * FROM spr.nimm__wgruppe( '-----This is an Array.', '2', '1' );

-- Modus „Einlesen\:
-- Die aktuelle Wortgruppe konnte nicht im System gefunden werden. Sie wird neu
angelegt.
SELECT * FROM spr.zeige__wgruppe;
SELECT * FROM spr.nimm__wgruppe( 'Das ist ein Array.', '1', '0' );
SELECT * FROM spr.zeige__wgruppe;

-- Modus „Einlesen\:
-- Die aktuelle Wortgruppe konnte im System gefunden werden.
SELECT * FROM spr.nimm__wgruppe( 'Das ist ein Array.', '1', '0' );
SELECT * FROM spr.zeige__wgruppe;
SELECT * FROM spr.nimm__wgruppe( 'Das ist ein Array.', '1', '0' );
SELECT * FROM spr.zeige__wgruppe;

-- Modus „Übersetzen\:
-- Die aktuelle Wortgruppe konnte nicht im System gefunden werden. Die Übersetzungsvariant
wird neu angelegt.
SELECT * FROM spr.nimm__wgruppe( 'Das ist ein Array.', '1', '0' );
SELECT * FROM spr.zeige__wgruppe;
SELECT * FROM spr.nimm__wgruppe( 'This is an Array.', '2', '1' );
SELECT * FROM spr.zeige__wgruppe;

-- Modus „Übersetzen\:
-- Die aktuelle Wortgruppe konnte im System gefunden werden.
SELECT * FROM spr.nimm__wgruppe( 'Das ist ein Array.', '1', '0' );
SELECT * FROM spr.nimm__wgruppe( 'This is an Array.', '2', '1' );
```

```
SELECT * FROM spr.zeige__wgruppe;
SELECT * FROM spr.nimm__wgruppe( 'This is an Array.', '2', '1' );
SELECT * FROM spr.zeige__wgruppe;
```

9.4 spr.zeige__wgelemente

Beschreibung der Spalten

spr_uebvar_id siehe spr.uebvar.id

spr_sprachen_id siehe spr.sprachen.id

spr_wgruppe_id siehe spr.wgruppe.id

wgelement siehe intern.abczk.id ODER intern.restzk.id

Beschreibung der Sicht

Es werden die Elemente der Wortgruppen zurückgegeben.

9.5 spr.zeige__wgruppe

Beschreibung der Spalten

spr_uebvar_id siehe spr.zeige__wgelemente.spr_uebvar_id

spr_sprachen_id siehe spr.zeige__wgelemente.spr_sprachen_id

spr_wgruppe_id siehe spr.zeige__wgelemente.spr_wgruppe_id

wgelement siehe spr.zeige__wgelemente.wgelement

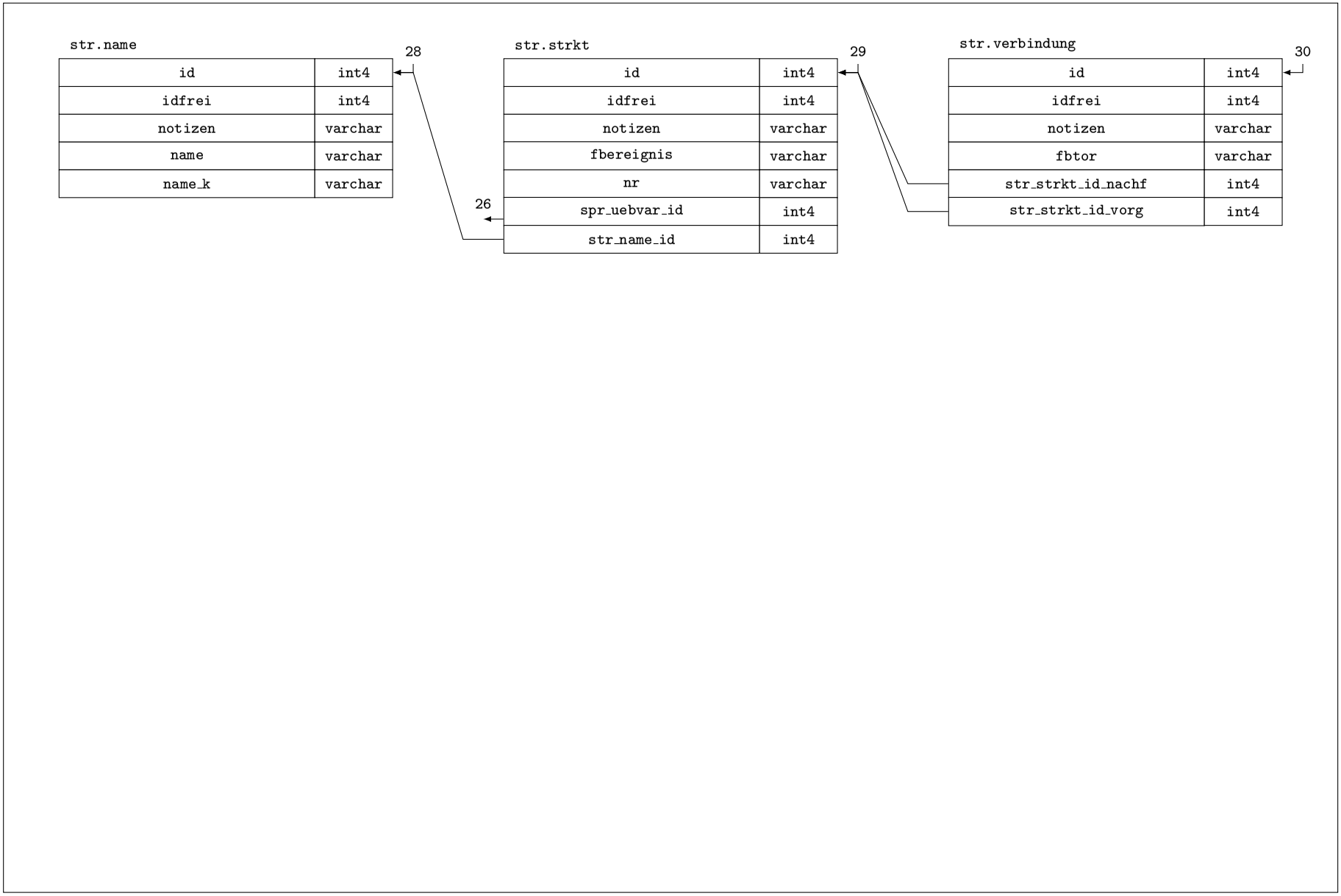
Beschreibung der Sicht

Es werden die vollständigen Wortgruppen zurückgegeben.

10 str

Das Schema „Strukturen“ dient der Verwaltung sämtlicher Strukturen (z.B. Fahrzeugstruktur, Gefahrennetz, Fehlerbaumstrukturen)

10.1 Tabellenhierarchie der Ebenen 1 — 3



10.2 Schlüssel und Bedingungen

Primärschlüssel

`pk__str_name_id` für Spalte `str.name.id`

`pk__str_strkt_id` für Spalte `str.strkt.id`

`pk__str_verbindung_id` für Spalte `str.verbindung.id`

Fremdschlüssel

`fk__str_strkt__spr_uebvar_id` aus Spalte `str.strkt.spr_uebvar_id`

`fk__str_strkt__str_name_id` aus Spalte `str.strkt.str_name_id`

`fk__str_verbindung__str_strkt_id_nachf` aus Spalte `str.verbindung.str_strkt_id_nachf`

`fk__str_verbindung__str_strkt_id_vorg` aus Spalte `str.verbindung.str_strkt_id_vorg`

Unique Constraints

`uc__str_name` in Tabelle `str.name` für die Spalte(n) `str.name.idfrei, str.name.name, str.name.name_k`

`uc__str_str_strkt` in Tabelle `str.strkt` für die Spalte(n) `str.strkt.idfrei, str.strkt.nr, str.strkt.spr_uebvar_id, str.strkt.str_name_id`

`uc__str_str_verbindung` in Tabelle `str.verbindung` für die Spalte(n) `str.verbindung.idfrei, str.verbindung.str_strkt_id_nachf, str.verbindung.str_strkt_id_vorg`

Check Constraints

`cc__str_verbindung` in Tabelle `str.verbindung` die Bedingung `(((str_strkt_id_nachf = 0) AND (str_strkt_id_vorg = 0)) OR ((id > 0) AND (str_strkt_id_vorg > 0)))`

10.3 str.nimm__strkt

Funktionsname

`str.nimm__strkt( IN tabwerte varchar[], OUT strktid varchar )`

Funktionsparameter/Rückgabewert(e)

`tabwerte` Liste der Werte, welche in `str.strkt` eingelesen werden sollen (Details siehe Funktionsbeschreibung — Werteliste einlesen). Sie werden in der folgenden Reihenfolge in die Spalten

1. `str_name_id`
2. `nr`
3. `spr_uebvar_id`
4. `fbereignis`

5. notizen

Es müssen mindestens die ersten drei Werte übergeben werden.

strktid Es wird die Identifikationsnummer des eingelesenen Strukturpunktes zurückgegeben (str.strkt.id)

Funktionsbeschreibung
Ziel der Funktion ist das Einlesen eines Strukturpunktes.

Werteliste einlesen:
Dem Funktionsparameter tabwerte ist eine Werteliste (ARRAY) zu übergeben (die maximale Anzahl und die Reihenfolge der Werte ist fest vorgegeben — siehe oben). Soll nur eine Teilmenge der Werte übergeben werden, dann sind alle Werte von links nach rechts zu spezifizieren (nichtbenötigte Werte am Ende der Liste dürfen weggelassen werden).

Es wird empfohlen, zum Einlesen des dritten Parameters die Funktion spr.nimm__wgruppe(...) zu verwenden (Hinweise zum Umgang siehe dort). Daher diese Funktion ein (ARRAY) zurückgibt, beim Einlesen jedoch nur der erste Wert benötigt wird, ist sie wie folgt aufzurufen:
varchar[(spr.nimm__wgruppe('Strukturpunkt ...', 'de_DE', NULL))[1]]

fbereignis - gültige Werte siehe str.pruefe__fbwerte(...)

Beispiele
```
-- Fehlermeldung. Die Mindestanzahl der Elemente muß stimmen.
SELECT * FROM str.nimm__strkt( ARRAY['1', '-BA'] );

-- Fehlermeldung. Der Parameter tabwerte[1] muß größer 0 sein.
SELECT * FROM str.nimm__strkt( ARRAY['0', '-BA', '11'] );

-- Fehlermeldung. Der Parameter tabwerte[4] muß gültige Werte enthalten.
SELECT * FROM str.nimm__strkt( ARRAY['1', '-BA', '11', 'Ungültig'] );

-- Strukturpunkt einlesen
SELECT * FROM str.strkt;
SELECT * FROM str.nimm__strkt( ARRAY['1', '-BAAA', spr.nimm__wgruppe( 'Strukturpunkt',
'1', '0' ), 'GE', 'interne Notizen']);
SELECT * FROM str.strkt;
```

10.4 str.nimm__verbindung

Funktionsname
str.nimm__verbindung(IN tabwerte varchar[], OUT verbid varchar)

Funktionsparameter/Rückgabewert(e)

`tabwerte` Liste der Werte, welche in `str.verbindung` eingelesen werden sollen (Details siehe Funktionsbeschreibung — Werteliste einlesen). Sie werden in der folgenden Reihenfolge in die Spalten

1. `str_strkt_id_vorg`
2. `str_strkt_id_nachf`
3. `fbtor`
4. `notizen`

Es müssen mindestens die ersten beiden Werte übergeben werden. Sie müssen größer `0` sein.

`verbid` Es wird die Identifikationsnummer der aktuell eingelesenen Verbindung der Strukturpunkte `str.verbindung.id` zurückgegeben.

Funktionsbeschreibung
Ziel der Funktion ist das Verbinden der Strukturpunkte und Zuordnen eines Fehlerbaumtores (Verknüpfung / engl. Gate).

Werteliste einlesen:
Dem Funktionsparameter `tabwerte` ist eine Werteliste (`ARRAY`) zu übergeben (die maximale Anzahl und die Reihenfolge der Werte ist fest vorgegeben — siehe oben). Soll nur eine Teilmenge der Werte übergeben werden, dann sind alle Werte von links nach rechts zu spezifizieren (nichtbenötigte Werte am Ende der Liste dürfen weggelassen werden).

`fbtor` - gültige Werte siehe `str.pruefe__fbwerte(...)`

Beispiele
```
-- Fehlermeldung. Die Mindestanzahl der Elemente muß stimmen.
SELECT * FROM str.nimm__verbindung( ARRAY['1'] );

-- Fehlermeldung. Die Parameter tabwerte[1] und tabwerte[1] müssen größer 0 sein.
SELECT * FROM str.nimm__verbindung( ARRAY['0', '1'] );
SELECT * FROM str.nimm__verbindung( ARRAY['1', '0'] );

-- Fehlermeldung. Der Parameter tabwerte[3] muß gültige Werte enthalten.
SELECT * FROM str.nimm__verbindung( ARRAY['1', '1', 'Ungültig'] );

-- Strukturpunkt einlesen
SELECT * FROM str.verbindung ORDER BY id DESC LIMIT 3;
SELECT * FROM str.verbindung ORDER BY id DESC LIMIT 3;
SELECT * FROM str.nimm__verbindung( ARRAY[str.nimm__strkt( ARRAY['1', '-BAAA', spr.nimm__wgr
'Strukturpunkt', '1', '0' )]), str.nimm__strkt( ARRAY['1', '-BAAA-1', spr.nimm__wgruppe(
'Strukturpunkt-1', '1', '0' )]), 'UND', 'interne Notizen'] );
SELECT * FROM str.verbindung ORDER BY id DESC LIMIT 3;
```

10.5 str.pruefe__fbwerte

Funktionsname
`str.pruefe__fbwerte( IN aspekt varchar, IN wert varchar, IN fktname varchar, IN paramname varchar )`

Funktionsparameter/Rückgabewert(e)

`aspekt` Sollen Fehlerbaum-Ereignisse geprüft werden, ist `FB-Ereignis` anzugeben. Sollen Fehlerbaum-Tore geprüft werden, ist `FB-Tor` anzugeben.

`wert` Es ist der Wert anzugeben, dessen Plausibiliät geprüft werden soll.

`fktname` Es ist der Name der Funktion anzugeben, welche die Prüfroutine aufruft. Der Funktionsname wird in der Fehlermeldung verwendet. Sollte er leer oder `NULL` sein, wird `str.pruefe__fbwerte(...)` verwendet.

`paramname` Es ist der Name des Parameters anzugeben, welcher den zu prüfenden Wert `wert` beinhaltet. Der Parametername wird in der Fehlermeldung verwendet. Sollte er leer oder `NULL` sein, wird `wert` verwendet.

Funktionsbeschreibung
Ziel der Funktion ist die Plausibilitätsprüfung gültiger Fehlerbaum-Ereignisse bzw. -Tore.

`FB-Ereignis` - gültige Werte nach DIN IEC 61025:1993-12):

EBB Ereignis-Beschreibungs-Block

GE Grundereignis

NUE Nicht weiter untersuchtes Ereignis

AUE Anderweitig untersuchtes Ereignis

HAUS Haus

NULL Null-Ereignis

VWVA Verweisungs-Zeichen von außen

VWNA Verweisungs-Zeichen nach außen

`FB-Tor` - gültige Werte nach DIN IEC 61025:1993-12):

UND UND-Verknüpfung

ODER ODER-Verknüpfung

XODER EXCLUSIV-ODER-Verknüpfung

NICHT NICHT-Verknüpfung

BED Bedingungs-Verknüpfung

RED Redundante Struktur

ALLG Verknüpfung (allgemein)

Beispiele

```
-- Fehlermeldung. Der übergebene Aspekt ist ungültig.
SELECT * FROM str.pruefe__fbwerte( NULL, 'HAUS', NULL, NULL );

-- Fehlermeldung. Der übergebene Wert darf Nicht NULL oder leer sein, bzw. muß
dem gültigen Wert entsprechen
SELECT * FROM str.pruefe__fbwerte( 'FB-Ereignis', NULL, NULL, NULL );
SELECT * FROM str.pruefe__fbwerte( 'FB-Ereignis', 'ungueltig', NULL, NULL );
SELECT * FROM str.pruefe__fbwerte( 'FB-Tor', 'ungueltig', 'fktname', 'fktparam'
);

-- Die Werte sind gültig
SELECT * FROM str.pruefe__fbwerte( 'FB-Ereignis', 'HAUS', NULL, NULL );
SELECT * FROM str.pruefe__fbwerte( 'FB-Tor', 'UND', NULL, NULL );
```

10.6 str.zeige__strkt

Beschreibung der Spalten

`str_name_id` siehe `str.name.id`

`str_name_name` siehe `str.name.name`

`str_name_name_k` siehe `str.name.name_k`

`str_name_notizen` siehe `str.name.notizen`

`str_strkt_id` siehe `str.strkt.id`

`str_strkt_nr` siehe `str.strkt.nr`

`str_strkt_spr_uebvar_id` siehe `str.strkt.spr_uebvar_id`

`str_strkt_fbereignis` siehe `str.strkt.fbereignis`

`str_strkt_notizen` siehe `str.strkt.notizen`

`str_verbindung_id` siehe `str.verbindung.id`

`str_strkt_id_nachf` siehe `str.verbindung.str_strkt_id_nachf`

`str_verbindung_fbtor` siehe `str.verbindung.fbtor`

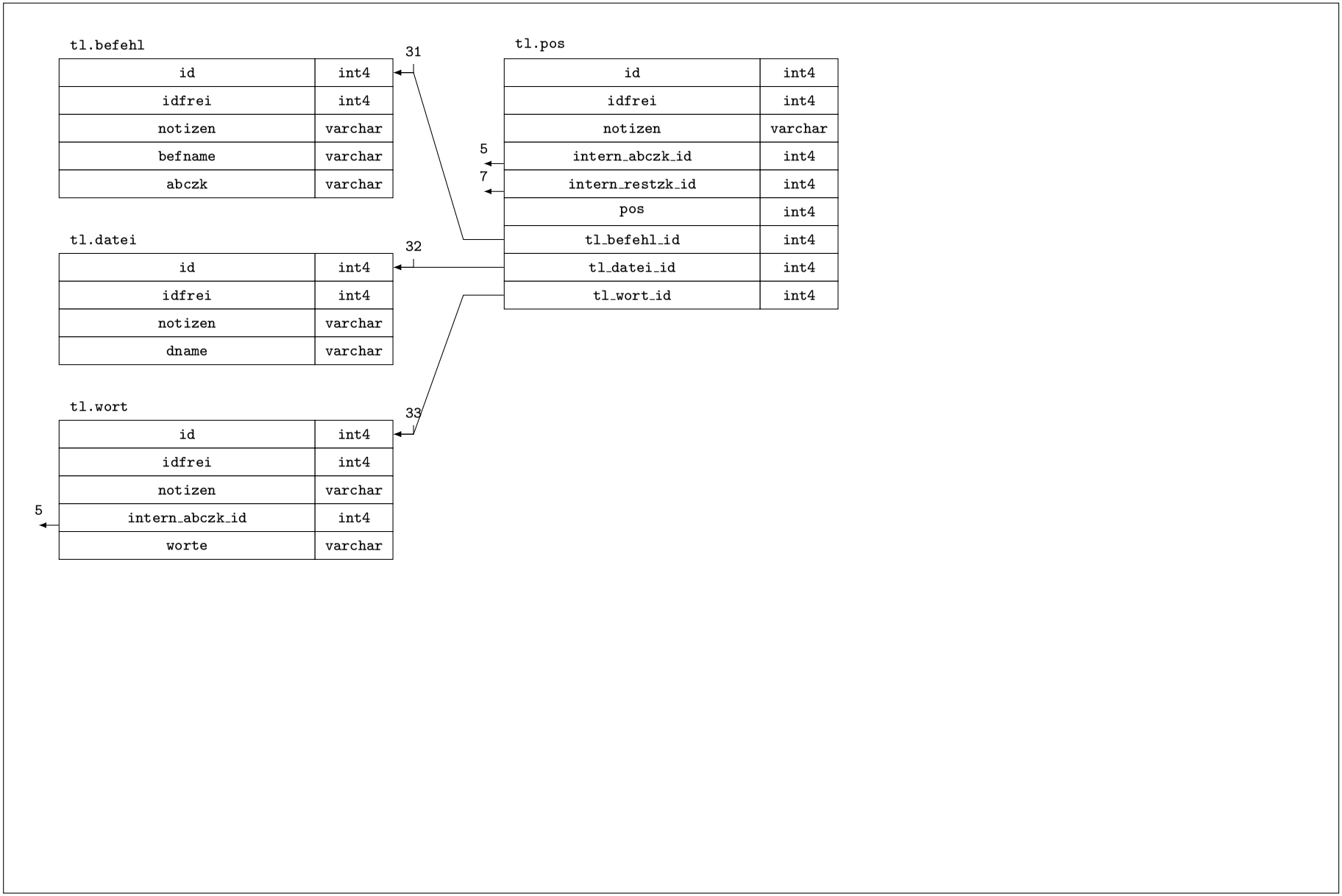
`str_verbindung_notizen` siehe `str.verbindung.notizen`

Beschreibung der Sicht
Es werden die Strukturpunkte (und deren Nachfolger) einschließlich zugeordneter Strukturbezeichnung zurückgegeben.

11 tl

Das Schema „Tex/Latex“ dient der Verwaltung der verwendeten TEX/LATEX - Befehle, TEX/LATEX - Wörter und sonstigen in TEX/LATEX - Dateien auftretenden Zeichenketten.

11.1 Tabellenhierarchie der Ebenen 1 — 3



11.2 Schlüssel und Bedingungen

Primärschlüssel

pk__tl_befehl_id für Spalte tl.befehl.id

pk__tl_datei_id für Spalte tl.datei.id

pk__tl_pos_id für Spalte tl.pos.id

pk__tl_wort_id für Spalte tl.wort.id

Fremdschlüssel

fk__tl_pos__intern_abczk_id aus Spalte tl.pos.intern_abczk_id

fk__tl_pos__intern_restzk_id aus Spalte tl.pos.intern_restzk_id

fk__tl_pos__tl_befehl_id aus Spalte tl.pos.tl_befehl_id

fk__tl_pos__tl_datei_id aus Spalte tl.pos.tl_datei_id

fk__tl_pos__tl_wort_id aus Spalte tl.pos.tl_wort_id

fk__tl_wort__intern_abczk_id aus Spalte tl.wort.intern_abczk_id

Unique Constraints

uc__tl_befehl in Tabelle tl.befehl für die Spalte(n) tl.befehl.idfrei, tl.befehl.befname, tl.befehl.abczk

uc__tl_datei in Tabelle tl.datei für die Spalte(n) tl.datei.idfrei, tl.datei.dname

uc__tl_pos in Tabelle tl.pos für die Spalte(n) tl.pos.idfrei, tl.pos.pos, tl.pos.tl_datei_id

uc__tl_wort in Tabelle tl.wort für die Spalte(n) tl.wort.idfrei, tl.wort.intern_abczk_id, tl.wort.worte

Check Constraints

cc__tl_pos in Tabelle tl.pos die Bedingung (((((((intern_abczk_id = 0) AND (intern_restzk_id = 0)) AND (tl_befehl_id = 0)) AND (tl_wort_id = 0)) OR ((((intern_abczk_id > 0) AND (intern_restzk_id = 0)) AND (tl_befehl_id = 0)) AND (tl_wort_id = 0))) OR ((((intern_abczk_id = 0) AND (intern_restzk_id > 0)) AND (tl_befehl_id = 0)) AND (tl_wort_id = 0))) OR ((((intern_abczk_id = 0) AND (intern_restzk_id = 0)) AND (tl_befehl_id > 0)) AND (tl_wort_id = 0))) OR ((((intern_abczk_id = 0) AND (intern_restzk_id = 0)) AND (tl_befehl_id = 0)) AND (tl_wort_id > 0)))

11.3 tl.erzeuge__abczk

Funktionsname
`tl.erzeuge__abczk( IN tlwort varchar, OUT abczk varchar )`

Funktionsparameter/Rückgabewert(e)

`tlwort` Es ist das Wort anzugeben, aus welchem die AbCZk extrahiert werden soll. ACHTUNG ! Das '\' ist doppelt anzugeben (zu maskieren). Bsp.: `tl.erzeuge__ABCZk( 'Änderung' )`

`abczk` Es wird die aus dem TL-Wort erzeugte AbCZk zurückgegeben.

Funktionsbeschreibung
Ziel der Funktion ist das Erzeugen einer AbCZk `abcZk` aus dem übergebenen TL-Wort `tlwort`. Es werden alle TL-Zweizeichenbefehle in `tlwort` mittels der in `tl.befehl.abczk` zugeordneten AbCZk ersetzt.
Sollte `tlwort` leer sein, oder beim Erzeugen ein Fehler auftreten bzw. `abcZk` kein AbCZk darstellen, wird eine Fehlermeldung generiert.

Beispiele
```
-- Fehlermeldung. Der Parameter tlwort darf nicht leer oder NULL sein.
SELECT * FROM tl.erzeuge__ABCZk( '' );

-- Fehlermeldung. Es konnten keine TL-Zweizeichenbefehle gefunden werden.
SELECT * FROM tl.erzeuge__ABCZk( 'Änderung' );

-- Änderung zurückgeben
SELECT intern.nimm__zeile( 'tl.befehl', ARRAY['abczk', 'befname'], ARRAY['Ä', 'Ä']
);
SELECT intern.nimm__zeile( 'tl.befehl', ARRAY['befname'], ARRAY[E''] );
SELECT * FROM tl.erzeuge__ABCZk( E'Änderung' );
```

11.4 tl.erzeuge__dbmodell

Funktionsname
`tl.erzeuge__dbmodell( OUT tldateiinhalt varchar )`

Funktionsparameter/Rückgabewert(e)

`tldateiinhalt` Es wird der Inhalt der TEX/LATEX-Datei zurückgegeben, welche das Datenbankmodell enthält.

Funktionsbeschreibung
Ziel der Funktion ist das Erzeugen einer TEX/LATEX-Datei, welche das Datenbankmodell enthält. Sie wird aus den Kommentaren erzeugt, welche den verschiedenen Datenbankobjekten zugeordnet wurden. Es werden die Kommentare der Datenbank, Schemas, Tabellen,

Spalten, Funktionen, Primär- und Fremdschlüssel, sowie die `unique` und `check constraints` verwendet.
In den Kommentaren sind \ entsprechend zu doppeln. Unterstrichen werden automatisch \ vorangestellt, weshalb zum momentanen Zeitpunkt keine mathematischen Formeln eingegeben werden können.

Hinweise zur Benutzung:
Um nicht auf eine separate Ausleseroutine angewiesen zu sein, wird folgende (in `psql`) auszufuehrende Handlungsweise empfohlen:

```
\pset tuples_only \pset format unaligned \pset footer
SELECT * FROM tl.erzeuge__DBModell() \o dateiname.tex
;
\q
```

Alternativ kann in der Kommandozeile
`psql -U micha vf -P tuples_only -P format=unaligned -P footer -c ‘‘SELECT * FROM tl.erzeuge__DBModell()‘‘ -o dateiname.tex`
eingegeben werden.

Beispiele

```
-- Datenbankmodell ausgeben
SELECT * FROM tl.erzeuge__DBModell();
```

11.5 tl.erzeuge__tldbglossary

Funktionsname

`tl.erzeuge__tldbglossary( IN begriff varchar, IN gloprfx varchar, IN bechreibung varchar, OUT tldbgl varchar )`

Funktionsparameter/Rückgabewert(e)

`begriff` Der im Textdokument und Glossar darzustellende Begriff (darf nicht leer sein).

`gloprfx` Im Glossar wird `gloPrfx` dem `begriff` vorangestellt (zur Positionierung im Glossar) - darf leer sein.

`bechreibung` Die im Glossar dem `begriff` zuzuordnende Beschreibung (darf leer sein).

`tldbgl` Der im TL-Dokument einzusetzende `\dbGlossary...` - Befehl.

Funktionsbeschreibung

Ziel der Funktion ist das Erzeugen eines `\dbGlossary...` - Befehls welcher im TL-Dokument verwendet werden kann.
Sollte `bechreibung` leer sein, wird ausschließlich `begriff` zurückgegeben.

Beispiele

```
-- Fehlermeldung. Der Parameter begriff darf nicht leer oder NULL sein.
SELECT * FROM tl.erzeuge__TLdbGlossary( NULL, '', '' );
SELECT * FROM tl.erzeuge__TLdbGlossary( '', '', '' );

-- Glossareintrag zurückgeben
SELECT * FROM tl.erzeuge__TLdbGlossary( 'cc__intern_alle', 'intern.alle', NULL );
SELECT * FROM tl.erzeuge__TLdbGlossary( 'cc__intern_alle', 'intern.alle', '' );
SELECT * FROM tl.erzeuge__TLdbGlossary( 'cc__intern_alle', 'intern.alle', 'Es sind
die Zeichenketten einzugeben.' );
```

11.6 tl.erzeuge__tlzk

Funktionsname
`tl.erzeuge__tlzk( IN zk varchar, OUT tlzk varchar )`

Funktionsparameter/Rückgabewert(e)

`zk` Es ist die in TEX/LATEX zu wandelnde Zeichenkette anzugeben.

`tlzk` Es wird die in TEX/LATEX gewandelte Zeichenkette zurückgegeben.

Funktionsbeschreibung
Ziel der Funktion ist das Umwandeln von Zeichenketen in TEX/LATEX-Zeichenketten, welche im Absatzmodus akzeptiert werden können.

Beispiele
```
-- Es wird eine Leerzeichenkette bzw. NULL zurückgegeben
SELECT * FROM tl.erzeuge__TLZk( '' );
SELECT * FROM tl.erzeuge__TLZk( NULL );

-- Es wird intern_abczk_id zurückgegeben
SELECT * FROM tl.erzeuge__TLZk( 'intern_abczk_id' );

-- Es wird normale Zk zurückgegeben
SELECT * FROM tl.erzeuge__TLZk( 'normale Zk' );
```

11.7 tl.gib__datei

Funktionsname
`tl.gib__datei( IN dateiname varchar, OUT dateiinhalt varchar )`

Funktionsparameter/Rückgabewert(e)

`dateiname` Vollständiger Dateiname (einschl. Dateierweiterung) der auszugebenden TL-Datei

`dateiinhalt` Vollständiger Inhalt der auszugebenden TL-Datei

Funktionsbeschreibung
Ziel der Funktion ist das Herausgeben des Inhalts der mittels `dateiname` spezifizierten TEX/LATEX-Quelltextdatei (TL-Datei).

Hinweise zur Benutzung:
Um nicht auf eine separate Ausleseroutine angewiesen zu sein, wird folgende (in `psql`) auszufuehrende Handlungsweise empfohlen:

```
\pset tuples_only
\pset format unaligned
\pset footer
SELECT * FROM tl.gib__datei( 'dateiname.tex' ) \o dateiname.tex
;
\q
```

Alternativ kann in der Kommandozeile
`psql -U micha vf -P tuples_only -P format=unaligned -P footer -c ``SELECT * FROM tl.gib__datei( 'dateiname.tex' );`` -o dateiname.tex`
eingegeben werden.

Beispiele
```
-- Fehlermeldung. Der Parameter dateiname darf nicht leer oder NULL sein.
SELECT * FROM tl.gib__datei( NULL );

-- Dateiinhalt einlesen und wieder ausgeben
SELECT intern.nimm__zeile( 'tl.befehl', ARRAY['abczk', 'befname'], ARRAY['Ä', '
Ä'] );
SELECT intern.nimm__zeile( 'tl.befehl', ARRAY['befname'], ARRAY[E'\-'] );
SELECT intern.nimm__zeile( 'tl.datei', ARRAY['dname'], ARRAY['dateiname.tex'] );
SELECT * FROM tl.nimm__datei(
E'Än\-de\-rung
Än\-de\-rungHo\-sen\-bein
'' 2006-01-01 #1 Änderung
norm-dt_01050_din_en
\befehl'
, 'dateiname.tex' );
SELECT * FROM tl.gib__datei( 'dateiname.tex' );
```

11.8 tl.gib__ere_tlbefehl

Funktionsname
`tl.gib__ere_tlbefehl( OUT ere_tlbefehl varchar )`

Funktionsparameter/Rückgabewert(e)

`ere_tlbefehl` Ist der erweiterte regulaere Ausdruck zur Beschreibung eines TL-Befehls

Funktionsbeschreibung
Ziel der Funktion ist das Definieren und Zurueckgeben eines erweiterten regulaeren Ausdruckes (ERE), welcher das Textmuster eines TEX/LATEX - Befehls (TL-Befehl) beschreibt.
TL-Befehle beginnen mit einem oder @ oder Buchstaben dem ein Buchstabe folgen muß. TL-Befehle, welche mit einem Buchstaben beginnen, müssen mindestens ein - _ oder eine Ziffer enthalten.

TL-Zweizeichenbefehle werden über `tl.gib__ERE_TLZZBefehl()` abgefangen.
TL-Befehle, welche mit - _ oder einer Ziffer beginnen, werden bis zum ersten Buchstaben `intern.gib__ERE_RestZK()` abgebildet (TL-Befehl wird zerteilt).
Klammern und Sterne welche Teil eines TL-Befehls sein können, werden mittels `intern.gib__ERE_RestZK` abgebildet (TL-Befehl wird zerteilt).

Es ist besonders zu beachten, daß auch ABCZk (siehe `intern.gib__ERE_ABCZK()`) beschrieben werden, solange sie keine deutschen Sonderzeichen (ÄÖÜ, ...) enthalten ! Das ist von folgenden Routinen abzufangen !

Beispiele
```
-- ERE zurückgeben
SELECT * FROM tl.gib__ERE_TLBefehl();

-- Es wird test zurückgegeben
SELECT substring( 'test' from tl.gib__ERE_TLBefehl() );

-- Es wird \befehl zurückgeben
SELECT substring( '\befehl' from tl.gib__ERE_TLBefehl() );

-- Es wird @manual zurückgeben
SELECT substring( '@manual' from tl.gib__ERE_TLBefehl() );

-- Es wird abs2def zurückgeben
SELECT substring( 'abs2def' from tl.gib__ERE_TLBefehl() );

-- Es wird absdef2 zurückgeben
SELECT substring( 'absdef2' from tl.gib__ERE_TLBefehl() );

-- Es wird norm-dt_01050_din_en zurückgeben
SELECT substring( 'norm-dt_01050_din_en' from tl.gib__ERE_TLBefehl() );

-- Es wird eine Leerzeichenkette zurückgeben
SELECT substring( '\' from tl.gib__ERE_TLBefehl() );
```

```
-- Es wird eine Leerzeichenkette zurückgeben
SELECT substring( '\\' from tl.gib__ERE_TLBefehl() );

-- Es wird eine Leerzeichenkette zurückgeben
SELECT substring( '\@befehl' from tl.gib__ERE_TLBefehl() );

-- Es wird \bef zurückgeben (leider)
SELECT substring( '\bef@befehl' from tl.gib__ERE_TLBefehl() );

-- Es wird eine Leerzeichenkette zurückgeben
SELECT substring( '2absdef' from tl.gib__ERE_TLBefehl() );

-- Es wird eine Leerzeichenkette zurückgeben
SELECT substring( '-no-dt_01050_din_en' from tl.gib__ERE_TLBefehl() );

-- Es wird eine Leerzeichenkette zurückgeben
SELECT substring( '_no-dt_01050_din_en' from tl.gib__ERE_TLBefehl() );

-- Es wird eine Leerzeichenkette zurückgeben
SELECT substring( '2no-dt_01050_din_en' from tl.gib__ERE_TLBefehl() );
```

11.9 tl.gib__ere_tlwort

Funktionsname
`tl.gib__ere_tlwort( OUT ere_tlwort varchar )`

Funktionsparameter/Rückgabewert(e)

`ere_tlwort` Ist der erweiterte reguläre Ausdruck zur Beschreibung eines TL-Wortes.

Funktionsbeschreibung
Ziel der Funktion ist das Definieren und Zurückgeben eines erweiterten regulären Ausdruckes (ERE), welcher das Textmuster eines TL-Wortes beschreibt.
Ein TL-Wort beginnt mit einem Buchstaben oder mit einem ïn Kombination mit dem zulässigen Buchstaben.
In der Folge setzt es sich aus TL-Zweizeichenbefehlen (siehe `tl.gib__ERE_TLZZBefehl()`) und ABCZk (siehe `intern.gib__ERE_ABCZK()`) zusammen.
Es ist besonders zu beachten, daß auche ASCII-Wörter (siehe `intern.gib__ERE_ABCZK()`) beschrieben werden, solange sie keine deutschen Sonderzeichen (ÄÖÜ, ...) enthalten ! Das ist von folgenden Routinen abzufangen !

Beispiele

```
-- ERE zurückgeben
SELECT * FROM tl.gib__ERE_TLWort();
```

```
-- Es wird test zurückgegeben (leider)
SELECT substring( 'test' from tl.gib__ERE_TLWort() );

-- Es wird eine Leerzeichenkette zurückgegeben
SELECT substring( E'\Aistfalsch' from tl.gib__ERE_TLWort() );

-- Es wird Äßn-de-rung zurückgegeben
SELECT substring( E'Äßnderung' from tl.gib__ERE_TLWort() );

-- Es wird Äßnde-rungä zurückgegeben
SELECT substring( E'Äßnderungä' from tl.gib__ERE_TLWort() );

-- Es wird ßnderungä zurückgegeben (leider)
SELECT substring( E'ßnderungä' from tl.gib__ERE_TLWort() );

-- Es wird Äßmann zurückgegeben
SELECT substring( 'Äßmann' from tl.gib__ERE_TLWort() );
```

11.10 tl.gib__ere_tlzzbefehl

Funktionsname
`tl.gib__ere_tlzzbefehl( IN istanfang bool, OUT ere_tlzzbefehl varchar )`

Funktionsparameter/Rückgabewert(e)

`istanfang` Der erweiterte reguläre Ausdruck des Anfangs eines TL-Zweizeichenbefehls soll zurückgegeben werden

`ere_tlzzbefehl` Ist der erweiterte reguläre Ausdruck zur Beschreibung eines TL-Zweizeichenbefehls.

Funktionsbeschreibung
Ziel der Funktion ist das Definieren und Zurückgeben eines erweiterten regulären Ausdruckes (ERE), welcher das Textmuster eines TL-Zweizeichenbefehls beschreibt. Die folgenden TL-Zweizeichenbefehle werden vom ERE erfaßt:

- Ä
- ä
- Ö
- ö
- Ü
- ü
-

-

Beispiele

```
-- ERE zurückgeben
SELECT * FROM tl.gib__ERE_TLZZBefehl( true );
SELECT * FROM tl.gib__ERE_TLZZBefehl( false );

-- Leerzeichenkette zurückgeben
SELECT substring( E'' from tl.gib__ERE_TLZZBefehl( true ) );

-- ü zurückgeben
SELECT substring( 'ü' from tl.gib__ERE_TLZZBefehl( true ) );

--  zurückgeben
SELECT substring( E'' from tl.gib__ERE_TLZZBefehl( false ) );
```

11.11 tl.gib__liste_tlzzbefehl

Funktionsname

```
tl.gib__liste_tlzzbefehl( IN tlwort varchar, OUT listetlzzbefehl varchar[] )
```

Funktionsparameter/Rückgabewert(e)

`tlwort` Es ist das Wort anzugeben, aus welchem die TL-Zweizeichenbefehle extrahiert werden sollen. ACHTUNG ! Das '\' ist doppelt anzugeben (zu maskieren). Bsp.: `tl.gib__Liste_TLZZBefehl( 'Änderung' )`

`listetlzzbefehl` Es wird die Liste der gefundenen TL-Zweizeichenbefehle als `ARRAY` vom Typ `VARCHAR` zurückgegeben. Sollte kein TL-Zweizeichenbefehl in `tlwort` enthalten oder `tlwort` leer sein, wird `NULL` zurückgegeben.

Funktionsbeschreibung

Ziel der Funktion ist das Extrahieren von TL-Zweizeichenbefehlen aus einem Wort. Das Wort besteht ausschließlich aus Buchstaben und (ggf.) aus TL-Zweizeichenbefehlen. Es wird nicht geprüft, ob die extrahierten TL-Zweizeichenbefehle im System vorhanden sind.

Beispiele

```
-- Es wird NULL zurückgegeben
SELECT * FROM tl.gib__Liste_TLZZBefehl( NULL );

-- Es wird NULL zurückgegeben
SELECT * FROM tl.gib__Liste_TLZZBefehl( '' );

-- Es wird NULL zurückgegeben
```

```
SELECT * FROM tl.gib__Liste_TLZZBefehl( 'Änderung' );

-- Es wird Ä zurückgegeben
SELECT * FROM tl.gib__Liste_TLZZBefehl( 'Änderung' );

-- Es wird Ä  zurückgegeben
SELECT * FROM tl.gib__Liste_TLZZBefehl( E'Änderung' );

-- Es wird  zurückgegeben
SELECT * FROM tl.gib__Liste_TLZZBefehl( E'Änderung' );
```

11.12 tl.ist__tlwort

Funktionsname
`tl.ist__tlwort( IN tlwort varchar, OUT isttlwort bool )`

Funktionsparameter/Rückgabewert(e)

`tlwort` Es ist das zu testende Wort anzugeben. ACHTUNG ! Das '\' ist doppelt anzugeben (zu maskieren). Bsp.: `tl.ist__TLWort( 'Änderung' )`

`isttlwort` Wenn `istTLWort = false` dann ist `tlwort` kein TL-Wort. Wenn `istTLWort = true` dann ist ein TL-Wort

Funktionsbeschreibung
Ziel der Funktion ist das Prüfen, ob die übergebene Zeichenkette `tlwort` als TL-Wort interpretiert werden kann.
TL-Worte bestehen ausschließlich aus AbcZk in Kombination mit mindestens einem TL-Zweizeichenbefehl (Definition und Gültigkeit siehe `tl.gib__ERE_TLWort()` unter der Bedingung, daß `intern.ist__abczk( tlwort ) = false` ist).

Beispiele
```
-- Es wird false zurückgegeben
SELECT * FROM tl.ist__TLWort( '' );

-- Es wird false zurückgegeben
SELECT * FROM tl.ist__TLWort( NULL );

-- Es wird false zurückgegeben
SELECT * FROM tl.ist__TLWort( 'Hosenbein' );

-- Es wird false zurückgegeben
SELECT * FROM tl.ist__TLWort( 'Änderung__' );

-- Es wird true zurückgegeben
```

```
SELECT * FROM tl.ist__TLWort( 'Änderung' );

-- Es wird true zurückgegeben
SELECT * FROM tl.ist__TLWort( 'Änderung' );

-- Es wird true zurückgegeben
SELECT * FROM tl.ist__TLWort( 'Hosenbein' );
```

11.13 tl.nimm__datei

Funktionsname
`tl.nimm__datei( IN dateiinhalt varchar, IN dateiname varchar, OUT tldateiid varchar )`

Funktionsparameter/Rückgabewert(e)

`dateiinhalt` Vollständiger Inhalt der einzulesenden TL-Datei (auch Zusatzmodule wie `bibtex`)

`dateiname` Vollständiger Dateiname (einschl. Dateierweiterung) der einzulesenden TL-Datei

`tldateiid` Es wird die Identifikationsnummer des Namens der Datei zurückgegeben, deren Inhalt eingelesen wurde.

Funktionsbeschreibung
Ziel der Funktion ist das Einlesen des Inhalts der mittels `dateiname` spezifizierten TEX/LATEX-Quelltextdatei (TL-Datei). Auch Quelltextdateien von TL-Zusatzpaketen (z.B. `bibtex`) werden als TL-Dateien interpretiert. Die Funktion extrahiert aus dem übergebenen TL-Dateiinhalt die enthaltenen TL-Objekte (siehe `tl.nimm_erkannte__Liste_TLObjekte(...)`) und liest sie in `tl.pos` ein.
Das Konzept zum extrahieren der TL-Objekte kann der Beschreibung zu `tl.nimm_erkannte__Liste_TLObjekt` entnommen werden

Hinweise zur Benutzung:
Um nicht auf eine separate Einleseroutine angewiesen zu sein, wird folgende (in `psql` auszuführende Handlungsweise empfohlen):

1. Es ist sichergestellt, daß die einzulesende TL-Datei im TEX/LATEX fehlerfrei compiliert werden kann

2. Es ist eine Kopie der einzulesenen TL-Datei anzulegen

3. In der Kopie sind alle \
   mittels \\
   und alle '
   mittels ''
   zu ersetzen

4. Dem ersten Zeichen in der Kopie ist der SQL-Teilbefehl
   `SELECT * FROM tl.nimm__datei( E'`
   voranzustellen
5. Dem letzten Zeichen in der Kopie mit dem Dateinamen `dateiname.tex` ist der SQL-Teilbefehl
   `', 'dateiname.tex');`
   hizuzufügen
6. Im Postgres-Kommandozeilenprogramm `psql` ist mittels des Befehls
   `\i dateiname.tex`
   die in den beschriebenen Schritten modifizierte Kopie einzulesen

Beispiele

```
-- Fehlermeldung. Der Parameter dateiinhalt darf nicht leer oder NULL sein.
SELECT * FROM tl.nimm__datei( '', 'dateiname.tex' );

-- Fehlermeldung. Der Parameter dateiname darf nicht leer oder NULL sein.
SELECT * FROM tl.nimm__datei( 'ttt', NULL );

-- Fehlermeldung. Der Dateiname muß im System vorhanden sein.
SELECT * FROM tl.nimm__datei( 'ttt', 'Unbekannter Dateiname.tex' );

-- Dateiinhalt einlesen und wieder ausgeben
SELECT intern.nimm__zeile( 'tl.befehl', ARRAY['abczk', 'befname'], ARRAY['Ä', '
Ä'] );
SELECT intern.nimm__zeile( 'tl.befehl', ARRAY['befname'], ARRAY[E'\-'] );
SELECT intern.nimm__zeile( 'tl.datei', ARRAY['dname'], ARRAY['dateiname.tex'] );
SELECT * FROM tl.nimm__datei(
E'Än\-de\-rung
Än\-de\-rungHo\-sen\-bein
'' 2006-01-01 #1 Änderung
norm-dt_01050_din_en
\befehl'
, 'dateiname.tex' );
SELECT * FROM tl.gib__datei( 'dateiname.tex' );
```

11.14 tl.nimm__tlwort

Funktionsname

```
tl.nimm__tlwort( IN tlwort varchar, IN tlwortid_korr varchar, OUT tlwortid varchar
)
```

Funktionsparameter/Rückgabewert(e)

`tlwort` Einzulesendes TL-Wort. ACHTUNG ! Das '\' ist doppelt anzugeben (zu maskieren).
Bsp.: `tl.nimm__TLWort( 'Ho`
`-sen`
`-bein', true )`

`tlwortid_korr` Wird eine im System vorhandene `tl.wort.id` benannt, so wird das im System vorhandene TL-Wort überschrieben. Soll ein Wort hinzugefügt werden, so ist entweder '0' oder `NULL` anzugeben.

`tlwortid` Es wird die Identifikationsnummer des eingelesenen TL-Wortes zurueckgegeben.

Funktionsbeschreibung
Ziel der Funktion ist das Einlesen des übergebenen TL-Wortes `tlWort` in die Tabelle `tl.wort`. Das gültige Textmuster eines TL-Wortes kann der Funktion `tl.ist__TLWort(...)` entnommen werden.
Die im TL-Wort enthaltenen TL-Zweizeichenbefehle werden extrahiert, deren AbcZk-Bedeutung aus `tl.befehl.abczk` ausgelesen und im TL-Wort an Stelle der TL-Zweizeichenbefehle eingesetzt. Die auf diese Weise erzeugte AbcZk wird in `intern.abczk` eingelesen und mit dem in `tl.wort` eingelesenen TL-Wort verknüpft.
Wird eine mittels `tlWortId_korr` übergebene `tl.wort.id` im System gefunden, so wird das im System vorhandene TL-Wort überschrieben. Andernfalls wird `tlWort` mit der vorgegebenen `tlWortId_korr` im System eingefügt.

Beispiele
```
-- Fehlermeldung. Es liegt kein TL-Wort vor
SELECT * FROM tl.nimm__TLWort( '', NULL );
SELECT * FROM tl.nimm__TLWort( 'Hosenbain', NULL ); -- Fehlermeldung

-- Test
SELECT intern.nimm__zeile( 'tl.befehl', ARRAY['abczk', 'befname'], ARRAY['Ä', 'Ä']
);
SELECT intern.nimm__zeile( 'tl.befehl', ARRAY['befname'], ARRAY[E''] );

-- id automatisch zuordnen
SELECT * FROM tl.nimm__TLWort( 'Änderung', NULL );

-- neuer Eintrag mit id=99
SELECT * FROM tl.nimm__TLWort( 'Hosenbein', '99' );

-- vorhandene id=99 überschreiben
SELECT * FROM tl.nimm__TLWort( 'Hosenbeine', '99' );

-- Test
SELECT * FROM tl.wort;
SELECT * FROM intern.abczk WHERE abczk = 'Änderung';
SELECT * FROM intern.abczk WHERE abczk = 'Hosenbeine';
```

11.15 tl.nimm_erkannte__liste_tlobjekte

Funktionsname
`tl.nimm_erkannte__liste_tlobjekte( IN dateiinhalt varchar, OUT tabname varchar, OUT tabnameid varchar )`

Funktionsparameter/Rückgabewert(e)

`dateiinhalt` Vollständiger Inhalt der einzulesenden TL-Datei

`tabname` Name der Tabelle, in die das erkannte TL-Objekt eingelesen wurde

`tabnameid id` des erkannten TL-Objektes

Funktionsbeschreibung
Ziel der Funktion ist das Zerlegen des Inhaltes `dateiinhalt` einer TL-Datei in TL-Objekte mittels Erweiterter Regulaerer Ausdruecke (ERE). Die erkannten TL-Objekte werden bei Bedarf in die Datenbank eingelesen.

Aus dem übergebenen TL-Dateiinhalt extrahiert die Funktion

- AbcZk welche bei Bedarf in `intern.abczk` (Gueltigkeit siehe `intern.gib__ERE_ABCZK()`),
- RestZk welche bei Bedarf in `intern.restzk` (Gueltigkeit siehe `intern.gib__ERE_RestZK()`),
- TL-Befehle welche bei Bedarf in `tl.befehl` (Gueltigkeit siehe `tl.gib__ERE_TLBefehl()`) und
- TL-Worte welche bei Bedarf in `tl.wort` (Gueltigkeit siehe `tl.gib__ERE_TLWort()`) eingelesen werden.

Die im TL-Wort auftretenden TL-Zweizeichenbefehle müssen im System vorhanden sein.

Sämtliche Zeichenketten, welche nicht diesen Kategorien zuordenbar sind, werden zerlegt und dann entsprechend eingelesen. Zum Beispiel werden TL-Variablennamen, welche ausschliesslich aus Buchstaben bestehen, in `intern.abczk` und nicht in `tl.befehl` eingelesen. Die Ursache liegt in der Unfähigkeit der Funktion, eine Kontextprüfung durchzufuehren.

Mit dem Verzicht des Einlesens von Wortgruppen (siehe `spr.wgruppe`) müssen mehrsprachige Texte mittels TL-Paketen gesetzt werden (z.B. `babel`). Ursache ist auch hier, die Unfähigkeit eine Kontextprüfung durchzuführen.

Beispiele
```
-- Es wird keine Zeile zurückgegeben und nichts eingelesen
SELECT * FROM tl.nimm_erkannte__Liste_TLObjekte( ' );
0
```

```
-- Es wird die Zeichenkette analysiert, die TL-Objekte eingelesen und zurückgegeben
SELECT * FROM tl.nimm_erkannte__Liste_TLObjekte( 'Änderung ÄnderungHosenbein 2006-01-01
#1 Änderung norm-dt_01050_din_en \befehl' );
```

A GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document freeïn the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of „copyleft“, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **„Document“**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **„you“**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **„Modified Version“** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **„Secondary Section“** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **„Invariant Sections“** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **„Cover Texts“** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **„Transparent“** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not „Transparent“ is called **„Opaque“**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **„Title Page“** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, „Title Page“ means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **„Entitled XYZ“** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **„Acknowledgements“**, **„Dedications“**, **„Endorsements“**, or **„History“**.) To **„Preserve the Title“** of such a section when you modify the Document means that it remains a section „Entitled XYZ“ according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other

conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled „History“, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled „History“ in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the „History“ section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled „Acknowledgements“ or „Dedications“, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled „Endorsements“. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled „Endorsements“ or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled „Endorsements“, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled „History“ in the various original documents, forming one section Entitled „History“; likewise combine any sections Entitled „Acknowledgements“, and any sections Entitled „Dedications“. You must delete all sections Entitled „Endorsements“.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an „aggregate“ if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled „Acknowledgements“, „Dedications“, or „History“, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
See http://www.gnu.org/copyleft/.
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License „or any later version“ applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled „GNU Free Documentation License“.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the „with...Texts.“ line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software

Verzeichnis der Tabellenspalten und Begriffe

Tabellenspalte / Begriff	Beschreibung	
dok.ausgabe	Tabelle zur Verwaltung der Ausgaben (Revisionsstände) des aktuellen Dokumentes. Die Revisionsstände werden mittels der Einträge der Spalte dok.ausgabe.stand unterschieden.	12
dok.ausgabe.datum	Datum der aktuellen Ausgabe des Dokumentes.	12
dok.ausgabe.dok_doku_id	Identifikationsnummer des aktuellen Dokumentes.	12
dok.ausgabe.istfrei	true — Das Dokument ist in der aktuellen Ausgabe freigegeben (nicht mehr im Entwurfsstand).	12
dok.ausgabe.stand	Revisionstand (Kennzeichen der Ausgabe) des aktellen Dokumentes.	12
dok.doku	Tabelle zur Verwaltung der Basisinformationen der Dokumente. Die Dokumente werden anhand des Eigentümers und der Dokumentennummer unterschieden.	12
dok.doku.intern_firma_id	Identifikationsnummer der Firma, welche Eigentümer des Dokumentes ist.	12
dok.doku.nr	Identifikationsnummer (Dokumentennummer) des Dokumentes.	12
dok.doku.spr_uebvar_id__kb	Identifikationsnummer der Übersetzungsvariante, welche die Kurzbeschreibung des Inhaltes (Zweckes, Zieles, ...) des Dokumentes darstellt.	12
dok.doku.spr_uebvar_id__ttl	Identifikationsnummer der Übersetzungsvariante, welche den Titel des Dokumentes (Dokumententitel) darstellt.	12
dok.komp	Mittels dieser Tabelle werden die Komponenten dem aktuellen Dokument zugeordnet, welche im direkten Zusammenhang stehen.	12
dok.komp.dok_doku_id	Identifikationsnummer des aktuellen Dokumentes.	12
dok.komp.kmp_komp_id	Identifikationsnummer der Komponente, welche dem aktuellen Dokument zugeordnet ist.	12
dok.strkt	Mittels dieser Tabelle werden die Strukturpunkte dem aktuellen Dokument zugeordnet, welche im direkten Zusammenhang stehen.	12
dok.strkt.dok_doku_id	Identifikationsnummer des aktuellen Dokumentes.	12

Tabellenspalte / Begriff	Beschreibung	
`dok.strkt.str_strkt_id`	Identifikationsnummer des Strukturpunktes, welcher dem aktuellen Dokument zugeordnet ist.	12
GNU Free Documentation License	*Der Zweck dieser Lizenz ist es, ein Handbuch, Textbuch oder ein anderes zweckdienliches und nützliches Dokument frei, im Sinne von Freiheit, zu machen; jedermann die Freiheit zu sichern, es zu kopieren und mit oder ohne Änderungen daran, sowohl kommerziell als auch nicht kommerziell weiter zu verbreiten. Weiterhin sichert diese Lizenz einem Autor oder Verleger die Möglichkeit, Anerkennung für seine Arbeit zu erhalten ohne für Änderungen durch Andere verantwortlich gemacht zu werden.* (siehe http://www.giese-online.de/gnufdl-de.html)	3, **3**
`intern.abczk`	Tabelle die alle Zeichenketten aufnimmt, welche ausschliesslich aus Buchstaben bestehen. Gross-/Kleinschreibung wird unterschieden.	19
`intern.abczk.abczk`	Es sind die Zeichenketten einzugeben.	19
`intern.alle`	Tabelle, welche die Grundstruktur für alle weiteren Tabellen bildet. Neben der Primaerschluesselspalte wird die Spalte zur Wiederverwendung freigegebener `id` und eine Spalte zur Aufnahme interner Notizen zur Verfügung gestellt.	19
`intern.alle.id`	Primärschlüsselspalte: Diese Spalte steht in allen Tabellen jedes Schemas an erster Stelle. In vielen Fällen wird auf sie mittels Fremdschlüssel referenziert.	19
`intern.alle.idfrei`	Hilfsspalte zur Id-Verwaltung: Diese Spalte folgt unmittelbar `id`. Sie wird mit dem aktuellen Wert in `id` versehen, wenn die aktuelle Zeile als gelöscht angesehen werden soll und die zugeordnete `id` für neue Einträge zur Verfügung steht, sonst wird `0` eingetragen.	19

Tabellenspalte / Begriff	Beschreibung	
intern.alle.notizen	Interne Notizen / Anmerkungen: Diese Spalte folgt unmittelbar idfrei. Sie dient der Aufnahme interner Notizen, die nicht in offiziellen Auswertungen Verwendung finden. Sie kann von allen Nutzern verwendet werden, die Werte in die Datenbank eingeben oder verändern.	19
intern.allecc__intern_alle	In die Tabelle intern.alle sollen keine Werte eingefügt werden.	21
intern.allewlog	Tabelle, welche die Grundstruktur für alle weiteren Tabellen mit Logbuchfunktion (Darstellen der Änderungshistorie) bildet. Neben den Informationen der Tabelle intern.alle wird die Uhrzeit des Eintrages, der Name des Benutzers, die Datenquelle (bei automatische eingelesenen Daten) und der Zustand, ob ein automatisch eingelesener Datensatz geprüft wurde, gehalten.	19
intern.allewlog.logdatuhrz	Es werden Datum und Uhrzeit des aktuellen Eintrages aufgenommen (als DEFAULT). Es wird die SQL-Funktion timeofday() verwendet welche sicherstellt, daß auch innerhalb einer Transaktion die aktuelle Zeit eingetragen wird.	19
intern.allewlog.logdqok	Es ist true anzugeben, wenn automatisch eingelesene Daten vom Spezialisten akzeptiert wurden.	19
intern.allewlog.logdquelle	Es ist die Datenquelle (bei automatisch eingelesenden Daten) des aktuellen Eintrags anzugeben.	19
intern.allewlog.logname	Es ist der Datenbankbenutzername anzugeben, welcher den aktuellen Eintrag im System hinzugefügt hat.	19
intern.allewlogcc__intern_allewlog	In die Tabelle intern.allewlog sollen keine Werte eingefügt werden.	21
intern.firma	Tabelle zur Verwaltung der in den Projekten referenzierten Firmen.	19
intern.firma.name	Es ist der vollständige Name der Firma anzugeben.	19
intern.firma.name_k	Es ist die Kurzbezeichnung des Namens der Firma anzugeben.	19
intern.restzk	Tabelle die alle Zeichenketten aufnimmt, welche nicht aus Buchstaben bestehen.	20
intern.restzk.restzk	Es sind die Zeichenketten einzugeben.	20

Tabellenspalte / Begriff	Beschreibung	
kmp.komp	In dieser Tabelle werden die im System verwendbaren Komponenten gehalten. Sie werden anhand des Herstellers und der Komponentennummer unterschieden.	43
kmp.komp.intern_firma_id	Identifikationsnummer des Herstellers der Komponente.	43
kmp.komp.nr	Komponentennummer vergeben vom Hersteller der Komponente.	43
kmp.komp.nrfzg	Komponentennummer vergeben vom Hersteller der Fahrzeuge.	43
kmp.komp.spr_uebvar_id	Komponentenbezeichnung festgelegt vom Hersteller der Komponente.	43
kmp.komp.spr_uebvar_id__fzg	Komponentenbezeichnung festgelegt vom Hersteller der Fahrzeuge.	43
kmp.strkt	In dieser Tabelle werden der aktuellen Komponente sämtliche Strukturaspekte zugeordnet (z.B. Einbauourt, Produktgruppe, Funktionsgruppe, firmanspezifische Struktur, Standardbezeichnung der Komponente)	43
kmp.strkt.kmp_komp_id	Datenbank-Identifikationsnummer der aktuellen Komponente.	43
kmp.strkt.str_strkt_id	Identifikationsnummer des Strukturaspektes, welcher der aktuellen Komponente zugeordnet ist.	43
prj.p_dokkomp	Diese Tabelle dient der projektabhängigen Zuordnung der Komponenten zu den Dokumenten.	50
prj.p_dokkomp.dok_komp_id	Identifikationsnummer der im Projekt gültigen Verbindung Dokument — Komponente.	50
prj.p_dokkomp.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	50
prj.p_dokstrkt	Diese Tabelle dient der projektabhängigen Zuordnung der Strukturpunkte zu den Dokumenten.	50
prj.p_dokstrkt.dok_strkt_id	Identifikationsnummer der im Projekt gültigen Verbindung Dokument - Strukturpunkt.	50
prj.p_dokstrkt.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	50
prj.p_kmpstrkt	Diese Tabelle dient der projektabhängigen Zuordnung der Strukturpunkte zu den Komponenten.	50
prj.p_kmpstrkt.kmp_strkt_id	Identifikationsnummer der im Projekt gültigen Verbindung Komponente — Strukturpunkt.	50

Tabellenspalte / Begriff	Beschreibung	
prj.p_kmpstrkt.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	50
prj.p_prtdokausgabe	Diese Tabelle dient der projektabhängigen Zuordnung der Dokumentausgaben zu den Protokolleinträgen / Sicherheitsanforderungen / Terminplaneinträgen.	50
prj.p_prtdokausgabe.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	50
prj.p_prtdokausgabe.prt_dokausgabe_id	Identifikationsnummer der im Projekt gültigen Verbindung PE / SF / TP — Strukturpunkt.	50
prj.p_prtkomp	Diese Tabelle dient der projektabhängigen Zuordnung der Komponenten zu den Protokolleinträgen / Sicherheitsanforderungen / Terminplaneinträgen.	51
prj.p_prtkomp.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	51
prj.p_prtkomp.prt_komp_id	Identifikationsnummer der im Projekt gültigen Verbindung PE / SF / TP — Komponente.	51
prj.p_prtstrkt	Diese Tabelle dient der projektabhängigen Zuordnung der Strukturpunkte zu den Protokolleinträgen / Sicherheitsanforderungen / Terminplaneinträgen.	51
prj.p_prtstrkt.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	51
prj.p_prtstrkt.prt_strkt_id	Identifikationsnummer der im Projekt gültigen Verbindung PE / SF / TP — Strukturpunkt.	51
prj.p_prtverbindung	Diese Tabelle dient der projektabhängigen gegenseitigen Zuordnung der Protokolleinträge / Sicherheitsanforderungen / Terminplaneinträge.	51
prj.p_prtverbindung.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	51
prj.p_prtverbindung.prt_verbindung_id	Identifikationsnummer der im Projekt gültigen Verbindung PE / SF / TP — PE / SF / TP	51
prj.personen	Tabelle zur Verwaltung der in den Projekten eingebundenen Personen.	50
prj.personen.intern_firma_id	Es ist die Identifikationsnummer der Firma anzugeben, in welcher die Person angestellt ist.	50
prj.personen.name_n	Es ist der Nachname der Person anzugeben.	50
prj.personen.name_v	Es ist der Vorname der Person anzugeben.	50

Tabellenspalte / Begriff	Beschreibung	
prj.personen.notizen	Es sind Zusatzinformationen (z.B. Position, Aufgabe, ...) zur Person anzugeben.	50
prj.pl_psgefabk	Mittels dieser Tabelle wird die projektabhängige Logbuchfunktion für ps.gefabk bereitgestellt.	52
prj.pl_psgefabk.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	52
prj.pl_psgefabk.ps_gefabk_id	Identifikationsnummer der im Projekt gültigen gefahrenszenariobezogenen Abkürzung.	52
prj.pl_psgefkomp	Mittels dieser Tabelle wird die projektabhängige Logbuchfunktion für ps.gefkomp bereitgestellt.	52
prj.pl_psgefkomp.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	52
prj.pl_psgefkomp.ps_gefkomp_id	Identifikationsnummer der im Projekt gültigen gefahrenszenariobezogenen Komponente.	52
prj.pl_psgefnetz	Mittels dieser Tabelle wird die projektabhängige Logbuchfunktion für ps.gefnetz bereitgestellt.	53
prj.pl_psgefnetz.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	53
prj.pl_psgefnetz.ps_gefnetz_id	Identifikationsnummer des im Projekt gültigen gefahrenszenariobezogenen Eintrages im Gefahrennetz.	53
prj.pl_psgefprot	Mittels dieser Tabelle wird die projektabhängige Logbuchfunktion für ps.gefprot bereitgestellt.	53
prj.pl_psgefprot.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	53
prj.pl_psgefprot.ps_gefprot_id	Identifikationsnummer des im Projekt gültigen gefahrenszenariobezogenen PE / SF / TE.	53
prj.pl_psgefstrkt	Mittels dieser Tabelle wird die projektabhängige Logbuchfunktion für ps.gefstrkt bereitgestellt.	54
prj.pl_psgefstrkt.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	54
prj.pl_psgefstrkt.ps_gefstrkt_id	Identifikationsnummer der im Projekt gültigen gefahrenszenariobezogenen Strukturpunktes.	54
prj.plu_psgefanm	Mittels dieser Tabelle wird die projektabhängige Logbuchfunktion für ps.gefanm bereitgestellt. Weiterhin können verschiedene Eigenschaften projektabhängig umbenannt werden.	54
prj.plu_psgefanm.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	54

Tabellenspalte / Begriff	Beschreibung	
prj.plu_psgefanm.ps_gefanm_id	Identifikationsnummer der im Projekt gültigen gefahrenszenariobezogenen Anmerkung.	54
prj.plu_psgefanm.spr_uebvar_id	Identifikationsnummer der projektabhängig angepaßten Spalte ps.gefanm.spr_uebvar_id.	54
prj.pp_dokausgabe	Diese Tabelle dient der projektabhängigen Zuordnung der Personen zu den Ausgabeständen der Dokumente.	55
prj.pp_dokausgabe.dok_ausgabe_id	Es ist die Datenbank-Identifikationsnummer des aktuelle Revisionsstandes des Dokumentes anzugeben.	55
prj.pp_dokausgabe.prj_personen_id	Es ist die Identifikationsnummer der für das Dokument verantwortlichen Person anzugeben.	55
prj.pp_dokausgabe.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	55
prj.ppu_kmpkomp	Diese Tabelle dient der projektabhängigen Zuordnung der Personen zu den Komponenten. Weiterhin können verschiedene Eigenschaften projektabhängig umbenannt werden.	55
prj.ppu_kmpkomp.kmp_komp_id	Identifikationsnummer der im Projekt gültigen Komponente.	55
prj.ppu_kmpkomp.nrfzg	Projektabhängige Anpassung der Spalte kmp.komp.nrfzg.	55
prj.ppu_kmpkomp.prj_personen_id	Identifikationsnummer der für die Komponente verantwortlichen Person.	55
prj.ppu_kmpkomp.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	55
prj.ppu_kmpkomp.spr_uebvar_id__fzg	Projektabhängige Anpassung der Spalte kmp.komp.spr_uebvar_id__fzg.	55
prj.ppu_prtprotokoll	Diese Tabelle dient der projektabhängigen Zuordnung der Personen zu den Protokolleinträgen / Sicherheitsanforderungen / Terminplaneinträgen. Weiterhin können verschiedene Eigenschaften projektabhängig umbenannt werden.	56
prj.ppu_prtprotokoll.datuhrz	Datum / Uhrzeit des Entstehens des PE oder der SF. Für TP: Datum / Uhrzeit des Meilensteines bzw. Startpunkt einer Aktivität, welche über einen Zeitraum reicht.	56
prj.ppu_prtprotokoll.duanfrage	Datum / Uhrzeit der Bitte um Antwort auf bzw. Stellungnahme zum aktuellen PE / SF / TP.	56
prj.ppu_prtprotokoll.erledigt	Wenn der aktuellen PE, SF, TP erledigt ist, wird true angegeben.	56

Tabellenspalte / Begriff	Beschreibung	
prj.ppu_prtprotokoll.intern	Wenn der aktuelle PE, SF, TP ausschließlich intern verwendet werden darf, ist `true` anzugeben.	56
prj.ppu_prtprotokoll.prj_personen_id	Identifikationsnummer der Person, welche mit dem aktuellen PE / SF / TP in direktem Zusammenhang steht. Wurde `prj.ppu_prtprotokoll.teilnehmer = true` gesetzt, war die Person am / beim Entstehen des PE / SF / TP beteiligt.	56
prj.ppu_prtprotokoll.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	56
prj.ppu_prtprotokoll.prt_protokoll_id	Identifikationsnummer des aktuellen PE / SF / TP.	56
prj.ppu_prtprotokoll.spr_uebvar_id	Identifikationsnummer der projektabhängig angepaßten Spalte `prt.protokoll.spr_uebvar_id`.	56
prj.ppu_prtprotokoll.teilnehmer	Wird `true` gesetzt, war die beteiligte Person am / beim Entstehen des PE / SF / TP beteiligt. Wird `false` gesetzt, wurde der aktuellen Person eine Aufgabe gestellt.	56
prj.ppu_prtprotokoll.termin	Spätester Zeitpunkt (Datum / Uhrzeit) der Erledigung des aktuellen PE, SF, TP. Für TP: Endpunkt einer Aktivität, welche über einen Zeitraum reicht.	56
prj.ppu_psgefszenario	Diese Tabelle dient der projektabhängigen Zuordnung der Personen zu den Gefahrenszenarien. Weiterhin können verschiedene Eigenschaften projektabhängig umbenannt werden.	56
prj.ppu_psgefszenario.prj_personen_id	Identifikationsnummer der im Projekt für das Gefahrenszenario verantwortlichen Person.	56
prj.ppu_psgefszenario.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	56
prj.ppu_psgefszenario.ps_gefszenario_id	Identifikationsnummer des im Projekt gültigen Gefahrenszenarios.	56
prj.ppu_psgefszenario.spr_uebvar_id	Identifikationsnummer der projektabhängig angepaßten Spalte `ps.gefszenario.spr_uebvar_id`.	56
prj.ppu_psgefszenario.spr_uebvar_id__ursp	Identifikationsnummer der projektabhängig angepaßten Spalte `ps.gefszenario.spr_uebvar_id__ursp`.	56
prj.ppu_strstrkt	Diese Tabelle dient der projektabhängigen Zuordnung der Personen zu den Strukturpunkten. Weiterhin können verschiedene Eigenschaften projektabhängig umbenannt werden.	57
prj.ppu_strstrkt.fbereignis	Projektabhängig angepaßte Spalte `str.strkt.fbereignis`.	57

Tabellenspalte / Begriff	Beschreibung	
prj.ppu_strstrkt.prj_personen_id	Identifikationsnummer der im Projekt für den Strukturpunkt verantwortlichen Person.	57
prj.ppu_strstrkt.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	57
prj.ppu_strstrkt.spr_uebvar_id	Projektabhängig angepaßte Spalte str.strkt.spr_uebvar_id.	57
prj.ppu_strstrkt.str_strkt_id	Identifikationsnummer des im Projekt gültigen Strukturpunktes.	57
prj.projekte	Tabelle zur Verwaltung der projektspezifischen Informationen	50
prj.projekte.makefile	Es ist der Quelltext des makefiles anzugeben, welches der Erzeugung sämtlicher projektspezifischer Dokumente dient.	50
prj.projekte.prjbeschr	Es ist die Projektbeschreibung anzugeben. Hier sollen Besonderheiten des Projetes erfaßt werden.	50
prj.projekte.prjname	Es ist die Bezeichnung (der Name) des aktuellen Projektes anzugeben.	50
prj.pu_psabk	Diese Tabelle dient dem projektabhängigen Umbenennen verschiedener Eigenschaften.	57
prj.pu_psabk.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	57
prj.pu_psabk.ps_abk_id	Identifikationsnummer der im Projekt gültigen Abkürzung.	57
prj.pu_psabk.spr_uebvar_id	Identifikationsnummer der projektabhängig angepaßten Spalte ps.abk.spr_uebvar_id.	57
prj.pu_psabk.spr_uebvar_id__l	Identifikationsnummer der projektabhängig angepaßten Spalte ps.abk.spr_uebvar_id__l.	57
prj.pu_strverbindung	Diese Tabelle dient der projektabhängigen gegenseitigen Zuordnung der Strukturpunkte. Weiterhin können verschiedene Eigenschaften projektabhängig umbenannt werden.	57
prj.pu_strverbindung.fbtor	Projektabhängig angepaßte Spalte str.verbindung.fbtor.	57
prj.pu_strverbindung.prj_projekte_id	Identifikationsnummer des aktuellen Projektes.	57
prj.pu_strverbindung.str_verbindung_id	Identifikationsnummer der im Projekt gültigen Verbindung Strukturpunkt — Strukturpunkt.	57
prt.dokausgabe	Anhand dieser Tabelle werden den PE, SF und TP (ps.protokoll) die Dokumente zugeordnet.	70
prt.dokausgabe.dok_ausgabe_id	Es ist die Datenbank-Identifikationsnummer des aktuelle Revisionsstandes des Dokumentes anzugeben.	70
prt.dokausgabe.prt_protokoll_id	Es ist die Identifikationsnummer des PE, SF, TP anzugeben.	70

Tabellenspalte / Begriff	Beschreibung	
prt.komp	Anhand dieser Tabelle werden den PE, SF und TP (ps.protokoll) die Komponenten zugeordnet.	70
prt.komp.kmp_komp_id	Es ist die Dantenbank-Identifikationsnummer der Komponente anzugeben.	70
prt.komp.prt_protokoll_id	Es ist die Identifikationsnummer des PE, SF, TP anzugeben.	70
prt.protokoll	Tabelle zur Verwaltung der PE, SF, TP.	70
prt.protokoll.spr_uebvar_id	Identifikationsnummer der Übersetzungsvariante, welche den aktuellen PE, SF, TP darstellt.	70
prt.protokoll.typ	Klassifizierung des PE, SF, TP. Die möglichen Werte sind prt.nimm__xxx() zu entnehmen.	70
prt.strkt	Anhand dieser Tabelle werden den PE, SF und TP (ps.protokoll) die Srukturpunkte zugeordnet.	70
prt.strkt.prt_protokoll_id	Es ist die Identifikationsnummer des PE, SF, TP anzugeben.	70
prt.strkt.str_strkt_id	Es ist die Identifikationsnummer des Strukturpunktes anzugeben.	70
prt.verbindung	Tabelle zur Verwaltung der Abhängigkeiten der PE, SF, TP.	70
prt.verbindung.prt_protokoll_id	Es ist die Identifikationsnummer des PE, SF, TP.	70
prt.verbindung.prt_protokoll_id_vb	Es ist die Identifikationsnummer des PE, SF, TP anzugeben, welcher der Vorgänger des aktuellen PE, SF, TP ist.	70
ps.abk	In dieser Tabelle werden alle im Schema ps verwendeten Abkürzungen gehalten. Sie werden anhand ihres Typs (ps.abk.typ) klassifiziert und anhand ihres Typs und der Kurzbezeichnung unterschieden. Somit darf die Kurzbezeichnung innerhalb eines Typs nicht mehrfach auftreten.	79
ps.abk.spr_uebvar_id	Es ist die Abkürzung anzugeben.	79
ps.abk.spr_uebvar_id__l	Es ist die Beschreibung der Abkürzung anzugeben.	79
ps.abk.typ	Es ist der Typ der aktuellen Abkürzung anzugeben. Die gültigen Typen können ps.nimm__xxx() entnommen werden.	79
ps.gefabk	In dieser Tabelle werden die verschiedenen Aspekte zur Produktsicherheit, welche mittels Abkürzung dargestellt werden, dem Gefahrenszenarios zugeordnet.	79
ps.gefabk.ps_abk_id	Es ist die Identifikationsnummer der Abkürzung anzugeben, welche dem aktuellen Gefahrenszenario zugeordnet ist.	79

Tabellenspalte / Begriff	Beschreibung	
ps.gefabk.ps_gefszenario_id	Es ist die Identifikationsnummer des aktuellen Gefahrenszenarios anzugeben.	79
ps.gefanm	In dieser Tabelle werden dem Gefahrenszenario besondere Anmerkungen zugeordnet.	79
ps.gefanm.ps_gefszenario_id	Es ist die Identifikationsnummer des aktuellen Gefahrenszenarios anzugeben.	79
ps.gefanm.spr_uebvar_id	Identifikationsnummer der Übersetzungsvariante, welche eine Anmerkung zum aktuellen Gefahrenszenario darstellt.	79
ps.gefkomp	In dieser Tabelle werden dem Gefahrenszenario die Komponenten zugeordnet, welche nicht in Verbindung mit den Sicherheitsanforderungen stehen.	79
ps.gefkomp.kmp_komp_id	Es ist die Identifikationsnummer der Komponenten anzugeben, welche dem aktuellen Gefahrenszenario zugeordnet ist.	79
ps.gefkomp.ps_gefszenario_id	Es ist die Identifikationsnummer des aktuellen Gefahrenszenarios anzugeben.	79
ps.gefnetz	In dieser Tabelle werden die Auswirkungen eines Gefahrenszenarios mittels Referenz auf str.strkt dargestellt. In str.strkt ist ein Baum oder ein Netz abgebildet, welches die Äbhängigkeiten (Kausalitäten) der verschiedenen Gefahren beschreibt.	79
ps.gefnetz.ps_gefszenario_id	Es ist die Identifikationsnummer des aktuellen Gefahrenszenarios anzugeben.	79
ps.gefnetz.str_strkt_id	Es ist die Identifikationsnummer des Strukturpunktes anzugeben, welcher die Auswirkung des aktuellen Gefahrenszenarios beschreibt.	79
ps.gefprot	Diese Tabelle dient der Zuordnung der Protokolleinträge, Sicherheitsanforderungen, Terminplaneinträge zum aktuellen Gefahrenszenario.	80
ps.gefprot.prt_protokoll_id	Es ist die Identifikationsnummer des Protokolleintrages anzugeben, welcher dem aktuellen Gefahrenszenario zugeordnet werden soll.	80
ps.gefprot.ps_gefszenario_id	Es ist die Identifikationsnummer des aktuellen Gefahrenszenarios anzugeben.	80

Tabellenspalte / Begriff	Beschreibung	
ps.gefstrkt	In dieser Tabelle werden dem Gefahrenszenario die Strukturpunkte zugeordnet, welche nicht in Verbindung mit den Sicherheitsanforderungen stehen.	80
ps.gefstrkt.ps_gefszenario_id	Es ist die Identifikationsnummer des aktuellen Gefahrenszenarios anzugeben.	80
ps.gefstrkt.str_strkt_id	Es ist die Identifikationsnummer des Strukturpunktes anzugeben, welcher dem aktuellen Gefahrenszenario zugeordnet ist.	80
ps.gefszenario	In dieser Tabelle wird die Identifikationsnummer die Beschreibung und der Ursprung des Gefahrenszenarios abgelegt. Diese Tabelle stellt die zentrale Verbindung aller Sicherheitsaspekte dar. Die Einträge in dieser Tabelle dürfen nicht modifiziert werden. Sollte während der Projektarbeit ein Szenario als nicht passend erkannt werden, so ist dieses Szenario zu schließen (Hinweis im Gefahrenprotoll einfügen) und ein passendes Szenario zu suchen oder neu anzulegen., 79	
ps.gefszenario.spr_uebvar_id	Es ist die Identifikationsnummer der Übersetzungsvariante anzugeben, welche das aktuelle Gefahrenszenario beschreibt.	79
ps.gefszenario.spr_uebvar_id__ursp	Es ist die Identifikationsnummer der Übersetzungsvariante anzugeben, welche den Ursprung des aktuellen Gefahrenszenarios beschreibt.	79
spr.pos	Tabelle, die die Position des Wortes in der Wortgruppe aufnimmt (die Wortgruppe bildet). Die Reihenfolge der Worte wird mittels spr.pos.id definiert.	89
spr.pos.intern_abczk_id	Es ist die Identifikationsnummer des aktuell verwendeten Wortes anzugeben.	89
spr.pos.intern_restzk_id	Es ist die Identifikationsnummer der Zeichenkette anzugeben, welche keine Buchstaben enthält.	89
spr.pos.spr_wgruppe_id	Es ist die Identifikationsnummer der Wortgruppe anzugeben, in der das aktuelle Wort enthalten ist.	89

Tabellenspalte / Begriff	Beschreibung	
spr.poscc__spr_pos	Wenn intern_abczk_id größer Null ist, muß intern_restzk_id Null sein (und umgekehrt). Sind intern_abczk_id oder intern_restzk_id größer Null, muß spr_wgruppe_id immer größer Null sein.	90
spr.sprachen	Tabelle zur Verwaltung der unterschiedlichen Sprachen.	89
spr.sprachen.sprache	Bezeichnung der Sprache, welche für Übersetzungen verwendet wird.	89
spr.sprachen.sprache__k	Kurzbezeichnung der Sprache.	89
spr.uebersetzung	Tabelle, die die Verbindung zwischen Sprache, Übersetzungsvariante und Wortgruppe herstellt. Innerhalb einer Übersetzungsvariante darf eine Sprache maximal einmal zugeordnet sein.	89
spr.uebersetzung.spr_sprachen_id	Es ist die Identifikationsnummer der aktuell verwendeten Sprache anzugeben.	89
spr.uebersetzung.spr_uebvar_id	Es ist die Identifikationsnummer der aktuell verwendeten Übersetzungsvariante anzugeben.	89
spr.uebersetzung.spr_wgruppe_id	Es ist die Identifikationsnummer der aktuell verwendeten Wortgruppe anzugeben.	89
spr.uebvar	Mittels dieser Tabelle werden die möglichen Übersetzungsvarianten bereitgestellt. Sie können von weiteren Tabellen verwendet werden.	89
spr.wgruppe	Tabelle zur Verwaltung der unterschiedlichen Wortgruppen.	89
str.name	Tabelle zur Verwaltung der Bezeichnung der unterschiedlichen verwendeten Strukturen.	95
str.name.name	Strukturbezeichnung (Produktgruppenstruktur, Funktionsgruppenstruktur, ...).	95
str.name.name_k	Strukturbezeichnung (Kurzbezeichnung).	95
str.strkt	Tabelle zur Verwaltung der Strukturpunkte einschließlich Klassifizierung im Sinne eines Fehlerbaumereignisses (Basis, Haus, ...).	95
str.strkt.fbereignis	Es ist die Bezeichnung des Fehlerbaumereignisses (Basis, Haus, ...) anzugeben, welches dem aktuellen Strukturpunkt zugeordnet ist	95

Tabellenspalte / Begriff	Beschreibung	
str.strkt.nr	Nummer (Position) des aktuellen Strukturpunktes in einer Struktur. Die Nummer kann aus Zeichen, Ziffern oder einer Zeichen-/Ziffernkombination bestehen.	95
str.strkt.spr_uebvar_id	Identifikationsnummer zur mehrsprachigen Darstellung der Bezeichnung des aktuellen Strukturpunktes.	95
str.strkt.str_name_id	Identifikationsnummer der Strukturbezeichnung, welcher der aktuelle Strukturpunkt zugeordnet ist.	95
str.verbindung	Tabelle zur Verwaltung der Abhängigkeiten der Strukturpunkte einschließlich Zuordnung eines Fehlerbaumtores (AND, OR, ...).	95
str.verbindung.fbtor	Es ist die Bezeichnung des Fehlerbaumtores (AND, OR, ...) anzugeben, welches die Verbindung zwischen dem Vorgänger und dem Nachfolger darstellt. Der Vorgänger ist dem Ausgang des Tores zugeordnet. Der Nachfolger ist ein Eingang des Tores. Es muß manuell sichergestellt werden, daß sämtliche Nachfolger dasselbe Tor bezogen auf einen gemeinsamen Vorgänger haben.	95
str.verbindung.str_strkt_id_nachf	Identifikationsnummer des Strukturpunktes, welcher auf einen weiteren Strukturpunkt verweist (diesem folgt).	95
str.verbindung.str_strkt_id_vorg	Identifikationsnummer des Strukturpunktes, welcher von einem weiteren Strukturpunkt referenziert wird.	95
str.verbindungcc__str_verbindung	Wenn id Null ist, müssen str_strkt_id_vorg und str_strkt_id_nachf Null sein. Sonst muß str_strkt_id_vorg immer größer Null sein.	96
tl.datei.dname	Dateiname der TL-Datei.	103
tl.pos	Tabelle zum Verwalten der Position der unterschiedelichen Zeichenketten, welche in einer TL-Datei auftreten.	103
tl.pos.intern_abczk_id	Es ist die Identifikationsnummer des Wortes (besteht ausschliesslich aus Buchstaben) anzugeben, welches in der aktuellen TL-Datei verwendet wird.	103
tl.pos.intern_restzk_id	Es ist die Identifikationsnummer der Zeichenkette (enthält keine Buchstaben) anzugeben, die in der aktuellen TL-Datei verwendet wird.	103

Tabellenspalte / Begriff	Beschreibung	
tl.pos.pos	Es ist die Position des TL-Objektes in der TL-Datei anzugeben.	103
tl.pos.tl_befehl_id	Es ist die Identifikationsnummer des TL-Befehls anzugeben, welcher in der aktuellen TL-Datei verwendet wird.	103
tl.pos.tl_datei_id	Es ist die Identifikationsnummer der TL-Datei anzugeben, welche die aktuellen Zeichenketten aufnimmt.	103
tl.pos.tl_wort_id	Es ist die Identifikationsnummer des TL-Wortes anzugeben, welches in der aktuellen TL-Datei verwendet wird.	103
tl.poscc__tl_pos	Alle Fremdschlüssel duerfen Null sein. Wenn ein jedoch Fremdschlüssel größer Null ist, müssen die anderen Null sein.	104