|  |
| --- |
| GIBMIT |
| KNW - Stored Procedures |
| Modul 128 |
|  |
| **Benjamin Jenni** |
| **06.02.2017** |

|  |
| --- |
|  |

Inhalt

[Namenskonvention 2](#_Toc482372799)

[Definitionen 2](#_Toc482372800)

[Datenbank-Migration 3](#_Toc482372801)

[Kurzbeschreibung 3](#_Toc482372802)

[Ein-/ Ausgabeparameter 3](#_Toc482372803)

[Aufrufbeispiele 3](#_Toc482372804)

[Code 3](#_Toc482372805)

[Orte auslagern 4](#_Toc482372806)

[Kurzbeschreibung 4](#_Toc482372807)

[Ein-/ Ausgabeparameter 4](#_Toc482372808)

[Aufrufbeispiele 4](#_Toc482372809)

[Code 4](#_Toc482372810)

[Benutzer erstellen 5](#_Toc482372811)

[Kurzbeschreibung 5](#_Toc482372812)

[Ein-/ Ausgabeparameter 5](#_Toc482372813)

[Aufrufbeispiele 5](#_Toc482372814)

[Code 5](#_Toc482372815)

[Lernende archivieren 6](#_Toc482372816)

[Kurzbeschreibung 6](#_Toc482372817)

[Ein-/ Ausgabeparameter 6](#_Toc482372818)

[Aufrufbeispiele 6](#_Toc482372819)

[Code 6](#_Toc482372820)

# Namenskonvention

Um die Datenbank lesbarer und einheitlicher zu gestalten, wird bei der Namensgebung überall das gleiche Schema angewendet.

## Definitionen

1. Alle Attributnamen und Tabellennamen beginnen werden klein geschrieben.
2. Primary Keys werden nach folgendem Muster benannt: „id\_name“.
3. Foreign Keys werden nach folgendem Muster benannt: „fk\_name“.
4. Stored Procedures werden nach folgendem Muster benannt: „sp\_name“.
5. Cursor werden nach folgendem Muster benannt: „cr\_name“.

# Datenbank-Migration

## Kurzbeschreibung

Diese Stored Procedure nimmt die veraltete Datenbank mit ihren wirren Bezeichnungen und falschen Typen und wandelt sie in eine neue Datenbank um, in welcher die Namen der Namenskonvention entsprechen. Die Daten werden zudem aus der alten Datenbank in die neue übernommen.

## Ein-/ Ausgabeparameter

## Aufrufbeispiele

Untenstehend findet sich eine Variante, wie man die Stored Procedure aufrufen kann.

CALL sp\_migration();

## Code

DROP PROCEDURE IF EXISTS `sp\_migration`;

CREATE PROCEDURE `sp\_migration`()

BEGIN

DROP DATABASE IF EXISTS `schoolinfo\_neu`;

/\* Define settings \*/

SET DEFAULT\_STORAGE\_ENGINE = InnoDB;

SET CHARACTER\_SET\_CLIENT = utf8;

SET CHARACTER\_SET\_RESULTS = utf8;

SET CHARACTER\_SET\_CONNECTION = utf8;

SET COLLATION\_SERVER = utf8\_unicode\_ci;

SET COLLATION\_DATABASE = utf8\_unicode\_ci;

SET COLLATION\_CONNECTION = utf8\_unicode\_ci;

SET SQL\_MODE = 'ALLOW\_INVALID\_DATES';

CREATE DATABASE `schoolinfo\_neu`;

/\* ----- Create tables ------ \*/

/\* Create table for classes \*/

CREATE TABLE `schoolinfo\_neu`.`klassen` (

`id\_klasse` INT(10) NOT NULL AUTO\_INCREMENT,

`id\_lehrer` INT(10) DEFAULT NULL,

`name` VARCHAR(50) NOT NULL,

`beschreibung` VARCHAR(255) NOT NULL,

PRIMARY KEY (`id\_klasse`),

INDEX `id\_lehrer` (`id\_lehrer`)

);

/\* Create table for companies \*/

CREATE TABLE `schoolinfo\_neu`.`lehrbetriebe` (

`id\_lehrbetrieb` INT(10) NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(100) NOT NULL,

`strasse` VARCHAR(100) DEFAULT NULL,

`nummer` VARCHAR(10) DEFAULT NULL,

`plz` VARCHAR(10) DEFAULT NULL,

`ort` VARCHAR(50) DEFAULT NULL,

`kanton` VARCHAR(50) DEFAULT NULL,

`land` VARCHAR(50) DEFAULT NULL,

PRIMARY KEY (`id\_lehrbetrieb`)

);

/\* Create table for orientations \*/

CREATE TABLE `schoolinfo\_neu`.`fachrichtungen` (

`id\_fachrichtung` INT(10) NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(50) NOT NULL,

PRIMARY KEY (`id\_fachrichtung`)

);

/\* Create table for students \*/

CREATE TABLE `schoolinfo\_neu`.`lernende` (

`id\_lernender` INT(10) NOT NULL AUTO\_INCREMENT,

`anrede` VARCHAR(25) NOT NULL,

`name` VARCHAR(50) NOT NULL,

`vorname` VARCHAR(50) NOT NULL,

`geschlecht` ENUM ('M', 'F') NOT NULL,

`fk\_klasse` INT(10) DEFAULT NULL,

`bm` BOOL NOT NULL,

`fk\_fachrichtung` INT(10) NOT NULL,

`fk\_lehrbetrieb` INT(10) DEFAULT NULL,

`strasse` VARCHAR(50) DEFAULT NULL,

`plz` VARCHAR(10) DEFAULT NULL,

`ort` VARCHAR(50) DEFAULT NULL,

PRIMARY KEY (`id\_lernender`),

FOREIGN KEY (`fk\_klasse`) REFERENCES `klassen` (`id\_klasse`),

INDEX `fk\_klasse` (`fk\_klasse`),

FOREIGN KEY (`fk\_fachrichtung`) REFERENCES `fachrichtungen` (`id\_fachrichtung`),

INDEX `fk\_fachrichtung` (`fk\_fachrichtung`),

FOREIGN KEY (`fk\_lehrbetrieb`) REFERENCES `lehrbetriebe` (`id\_lehrbetrieb`),

INDEX `fk\_lehrbetrieb` (`fk\_lehrbetrieb`)

);

/\* Create table for modules \*/

CREATE TABLE `schoolinfo\_neu`.`module` (

`id\_modul` INT(10) NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(50) NOT NULL,

`beschreibung` VARCHAR(255) DEFAULT NULL,

PRIMARY KEY (`id\_modul`)

);

/\* Create table for marks \*/

CREATE TABLE `schoolinfo\_neu`.`noten` (

`fk\_lernender` INT(10) NOT NULL,

`fk\_modul` INT(10) NOT NULL,

`note\_knw` DECIMAL(3, 2) NOT NULL,

`note\_erfahrung` DECIMAL(3, 2) NOT NULL,

`datum\_erfahrungsnote` DATETIME DEFAULT NULL,

`datum\_knw` DATETIME DEFAULT NULL,

FOREIGN KEY (`fk\_lernender`) REFERENCES lernende (`id\_lernender`)

ON DELETE CASCADE,

INDEX `fk\_lernender` (`fk\_lernender`),

FOREIGN KEY (`fk\_modul`) REFERENCES module (`id\_modul`),

INDEX `fk\_modul` (`fk\_modul`)

);

/\* ----- Create column for old PKs ----- \*/

ALTER TABLE `schoolinfo\_neu`.`klassen`

ADD COLUMN id\_old INT(10) NOT NULL;

ALTER TABLE `schoolinfo\_neu`.`fachrichtungen`

ADD COLUMN id\_old INT(10) NOT NULL;

ALTER TABLE `schoolinfo\_neu`.`lehrbetriebe`

ADD COLUMN id\_old INT(10) NOT NULL;

ALTER TABLE `schoolinfo\_neu`.`lernende`

ADD COLUMN id\_old INT(10) NOT NULL;

ALTER TABLE `schoolinfo\_neu`.`module`

ADD COLUMN id\_old INT(10) NOT NULL;

ALTER TABLE `schoolinfo\_neu`.`noten`

ADD COLUMN id\_old\_lernender INT(10) NOT NULL,

ADD COLUMN id\_old\_modul INT(10) NOT NULL;

/\* ----- Copy Data ----- \*/

/\* Migrate orientations \*/

INSERT INTO `schoolinfo\_neu`.`fachrichtungen` (`id\_old`, `name`)

SELECT

`idrichtung` AS `id\_old`,

`richtung` AS `name`

FROM `schoolinfo1282017`.`richtung`;

INSERT INTO `schoolinfo\_neu`.`klassen` (id\_old, id\_lehrer, name, beschreibung)

SELECT

`idklasse` AS `id\_old`,

`klassenlehrer` AS `id\_lehrer`,

`name` AS `name`,

`realname` AS `beschreibung`

FROM `schoolinfo1282017`.`klasse`;

/\* Migrate companies \*/

INSERT INTO `schoolinfo\_neu`.`lehrbetriebe` (`id\_old`, `name`, `strasse`, `nummer`, `plz`, `ort`, `kanton`, `land`)

SELECT

`id\_Lehrbetrieb` AS `id\_old`,

`FName` AS `name`,

`FStrasse` AS `strasse`,

`FHausNr` AS `nummer`,

`FPlz` AS `plz`,

`FOrt` AS `ort`,

`FKanton` AS `kanton`,

IF(`FLand` = 'Schweiz', 'CH', `FLand`) AS `land`

FROM `schoolinfo1282017`.`lehrbetriebe`;

UPDATE `schoolinfo\_neu`.`lehrbetriebe`

SET

`strasse` = IF(`strasse` = '', NULL, `strasse`),

`nummer` = IF(`nummer` = '', NULL, `nummer`),

`kanton` = IF(`kanton` = '', NULL, `kanton`),

`land` = IF(`land` = '', NULL, `land`);

/\* Migrate classes \*/

INSERT INTO `schoolinfo\_neu`.`klassen` (`id\_old`, `id\_lehrer`, `name`, `beschreibung`)

SELECT DISTINCT

CONCAT('foobar-', `lernender`.`klasse`) AS `name`,

CONCAT('foobar-', `lernender`.`klasse`) AS `beschreibung`,

NULL AS `id\_lehrer`,

`lernender`.`klasse` AS `id\_old`

FROM `schoolinfo1282017`.`lernende` AS `lernender`

INNER JOIN `schoolinfo1282017`.`klasse` AS `k`

ON `k`.`idklasse` = `lernender`.`klasse`

WHERE `k`.`idklasse` IS NULL;

/\* Migrate students \*/

INSERT INTO `schoolinfo\_neu`.`lernende` (`id\_old`, `anrede`, `name`, `vorname`, `geschlecht`, `fk\_klasse`, `bm`, `fk\_fachrichtung`, `fk\_lehrbetrieb`, `strasse`, `plz`, `ort`)

SELECT

`lernender`.`Lern\_id` AS `id\_old`,

`lernender`.`anrede` AS `anrede`,

`lernender`.`name` AS `name`,

`lernender`.`vorname` AS `vorname`,

IF(`lernender`.`geschlecht` REGEXP '[Mm].\*', 'M', 'F') AS `geschlecht`,

`klassen`.`id\_klasse` AS `fk\_klasse`,

IF(`lernender`.`bm` = 0, FALSE, TRUE) AS `bm`,

`fachrichtungen`.`id\_fachrichtung` AS `fk\_fachrichtung`,

`lehrbetriebe`.`id\_Lehrbetrieb` AS `fk\_lehrbetrieb`,

IF(`lernender`.`strasse` = '', NULL, `lernender`.`strasse`) AS `strasse`,

IF(`lernender`.`plz` = '', NULL, `lernender`.`plz`) AS `plz`,

`lernender`.`ort` AS `ort`

FROM `schoolinfo1282017`.`lernende` AS `lernender`

LEFT JOIN `schoolinfo\_neu`.`klassen` ON `klassen`.`id\_old` = `lernender`.`klasse`

LEFT JOIN `schoolinfo\_neu`.`fachrichtungen` ON `fachrichtungen`.`id\_old` = `lernender`.`richtung`

LEFT JOIN `schoolinfo\_neu`.`lehrbetriebe` ON `lehrbetriebe`.`id\_old` = `lernender`.`lehrbetrieb`;

/\* Migrate modules \*/

INSERT INTO `schoolinfo\_neu`.`module` (`id\_old`, `name`, `beschreibung`)

SELECT

`idmodul` AS `id\_old`,

`m\_name` AS `name`,

`modulname` AS `beschreibung`

FROM `schoolinfo1282017`.`modul`;

/\* Migrate marks \*/

INSERT INTO `schoolinfo\_neu`.`noten` (`fk\_lernender`, `fk\_modul`, `note\_erfahrung`, `note\_knw`, `datum\_erfahrungsnote`, `datum\_knw`, `id\_old\_lernender`, `id\_old\_modul`)

SELECT

`lernender`.`id\_lernender` AS `fk\_lernender`,

`modul`.`id\_modul` AS `fk\_modul`,

`note`.`erfahrungsnote` AS `note\_erfahrung`,

`note`.`knw\_note` AS `note\_knw`,

`note`.`dat\_erfa` AS `datum\_erfahrungsnote`,

`note`.`dat\_knw` AS `datum\_knw`,

`lernender`.`id\_old` AS `id\_old\_lernender`,

`modul`.`id\_old` AS `id\_old\_modul`

FROM `schoolinfo1282017`.`noten` AS `note`

LEFT JOIN `schoolinfo\_neu`.`module` AS `modul` ON `modul`.`id\_old` = `note`.`module\_idmodule`

LEFT JOIN `schoolinfo\_neu`.`lernende` AS `lernender` ON `lernender`.`id\_old` = `note`.`lernende\_idLernende`

WHERE `lernender`.`id\_lernender` IS NOT NULL;

/\* ----- Delete old IDs ----- \*/

ALTER TABLE `schoolinfo\_neu`.`klassen`

DROP COLUMN `id\_old`;

ALTER TABLE `schoolinfo\_neu`.`fachrichtungen`

DROP COLUMN `id\_old`;

ALTER TABLE `schoolinfo\_neu`.`lehrbetriebe`

DROP COLUMN `id\_old`;

ALTER TABLE `schoolinfo\_neu`.`lernende`

DROP COLUMN `id\_old`;

ALTER TABLE `schoolinfo\_neu`.`module`

DROP COLUMN `id\_old`;

ALTER TABLE `schoolinfo\_neu`.`noten`

DROP COLUMN `id\_old\_lernender`,

DROP COLUMN `id\_old\_modul`;

END;

# Orte auslagern

## Kurzbeschreibung

Um die Datenbank zu normalisieren müssen unter anderem die Ortschaften ausgelagert werden. Dafür wird eine neue Tabelle erstellt, worauf dann die Lernenden und die Lehrbetriebe mit einem Fremdschlüssel verweisen.

## Ein-/ Ausgabeparameter

Ein- und Ausgabeparameter werden nicht benötigt, beziehungsweise sind nicht vorhanden.

## Aufrufbeispiele

Folgende Codezeile zeigt, wie man die Stored Procedure aufrufen kann.

CALL sp\_places();

## Code

# Benutzer erstellen

## Kurzbeschreibung

Es sollen neue Benutzer erstellt werden können. Mit dieser Stored Procedure werden Werte verarbeitet und abgespeichert.

## Ein-/ Ausgabeparameter

|  |  |  |
| --- | --- | --- |
| Parameter | Datentyp | Beschreibung |
| benutzer | VARCHAR(255) | Name des neuen Benutzers |
| passwort | VARCHAR(255) | Passwort des neuen Benutzers |
| hostname | VARCHAR(255) | Hostname |
| berechtigung | VARCHAR(255) | Die Berechtigungen, die vergeben werden sollen |
| zugriffsort | VARCHAR(255) | Die Datenbank, Tabelle oder Attribute, für welche die Berechtigungen gelten sollen. |

## Aufrufbeispiele

CALL sp\_users("user","123456aA","127.0.0.1","READ,WRITE","schoolinfo\_neu");

## Code

# Lernende archivieren

## Kurzbeschreibung

Lernende, welche die Ausbildung beendet haben, sollen aus der Datenbank einfacher als bisher entfernt werden. Dafür werden sie mit dieser Stored Procedure in eine andere de-normalisierte Datenbank verschoben.

## Ein-/ Ausgabeparameter

|  |  |  |
| --- | --- | --- |
| Parameter | Datentyp | Beschreibung |
| Klasse | VARCHAR(255) | Name der Klasse |

## Aufrufbeispiele

Mit dem Code der hier ist, kann eine Klasse ausgewählt werden, wessen Lernende anschliessend archiviert werden.

CALL sp\_archives("IAP13v");

## Code