GIBMIT

KNW - Stored Procedures

Modul 128

Benjamin Jenni

15.05.2017

Inhalt

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

Parameter werden keine benötigt.

Aufrufbeispiele

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

```
CALL sp_migration();
```

Code

sp_migration.sql

```
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`              BOOLEAN         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

sp_places.sql

```
DROP PROCEDURE IF EXISTS `sp_places`;

CREATE PROCEDURE `sp_places`()
  BEGIN

    /* Declare variables */
    DECLARE id_ort INT(10);
    DECLARE name VARCHAR(50);
    DECLARE plz VARCHAR(10);
    DECLARE lernende_fertig BOOLEAN DEFAULT FALSE;
    DECLARE lehrbetriebe_fertig BOOLEAN DEFAULT FALSE;
    DECLARE cr_lernende CURSOR FOR SELECT
                                     `id_lernender`,
                                     `plz`,
                                     `ort`
                                   FROM `schoolinfo_neu`.`lernende`;
    DECLARE cr_lehrbetriebe CURSOR FOR SELECT
                                         `id_lehrbetrieb`,
                                         `plz`,
                                         `ort`
                                       FROM `schoolinfo_neu`.`lehrbetriebe`;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET lernende_fertig = TRUE;

    DROP TABLE IF EXISTS `schoolinfo_neu`.`ortschaften`;

    /* Create table */
    CREATE TABLE `schoolinfo_neu`.`ortschaften` (
      `id_ort` INT(10) AUTO_INCREMENT,
      `name`   VARCHAR(50),
      `plz`    VARCHAR(10),
      PRIMARY KEY (`id_ort`),
      UNIQUE KEY `name_plz` (
        `name`,
        `plz`
      )
    );

    /* Add foreign key columns */
```

```
    ALTER TABLE `schoolinfo_neu`.`lernende`
      ADD COLUMN `fk_ort` INT(10) DEFAULT NULL,
      ADD FOREIGN KEY (`fk_ort`) REFERENCES `schoolinfo_neu`.`ortschaften`
(`id_ort`);

    ALTER TABLE `schoolinfo_neu`.`lehrbetriebe`
      ADD COLUMN `fk_ort` INT(10) DEFAULT NULL,
      ADD FOREIGN KEY (`fk_ort`) REFERENCES `schoolinfo_neu`.`ortschaften`
(`id_ort`);

    OPEN cr_lehrbetriebe;

    /* Loop to fetch data */
    myloop: LOOP
      IF lehrbetriebe_fertig
      THEN
        BEGIN
          FETCH cr_lernende
          INTO id_ort,
            name,
            plz;
        END;
      ELSE
        BEGIN
          FETCH cr_lehrbetriebe
          INTO id_ort,
            name,
            plz;
        END;
      END IF;

      IF lernende_fertig
      THEN
        IF lehrbetriebe_fertig
        THEN
          BEGIN
            CLOSE cr_lernende;
            LEAVE myloop;
          END;
        ELSE
          BEGIN
            SET lehrbetriebe_fertig = TRUE;
            CLOSE cr_lehrbetriebe;
            OPEN cr_lernende;
            SET lernende_fertig = TRUE;
            ITERATE myloop;
          END;
        END IF;
      END IF;

      SET @lernende_fertig = lernende_fertig;

      IF plz IS NOT NULL
      THEN
        SET @id_ort = NULL;
        SELECT
          `id_ort`,
          `name`,
          `plz`
        INTO
          @id_ort,
```

```
        @name,
        @plz
      FROM `schoolinfo_neu`.`ortschaften`
      WHERE `name` = name AND `plz` = plz
      LIMIT 1;

      IF @id_ort IS NULL
      THEN
        BEGIN
          INSERT INTO `schoolinfo_neu`.`ortschaften` (
            `name`,
            `plz`
          ) VALUES (
            name,
            plz
          );
          SET @id_ort = LAST_INSERT_ID();
        END;
      END IF;

      IF lehrbetriebe_fertig
      THEN
        UPDATE `schoolinfo_neu`.`lernende`
        SET `id_ort` = @id_ort
        WHERE `id_lernender` = id_ort
        LIMIT 1;
      ELSE
        UPDATE `schoolinfo_neu`.`lehrbetriebe`
        SET `id_ort` = @id_ort
        WHERE `id_lehrbetrieb` = id_ort
        LIMIT 1;
      END IF;

      SET lernende_fertig = @lernende_fertig;
    END IF;
  END LOOP;

  /* Drop old columns */
  ALTER TABLE `schoolinfo_neu`.`lernende`
    DROP COLUMN `plz`,
    DROP COLUMN `ort`;

  /* Drop old columns */
  ALTER TABLE `schoolinfo_neu`.`lehrbetriebe`
    DROP COLUMN `plz`,
    DROP COLUMN `ort`;

END;
```

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(100)	Name des neuen Benutzers
passwort	VARCHAR(100)	Passwort des neuen Benutzers
hostname	VARCHAR(100)	Hostname
berechtigung	VARCHAR(100)	Die Berechtigungen, die vergeben werden sollen
zugriffsort	VARCHAR(100)	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");

CALL sp_users("foobar","fancy1b8","localhost","DROP","schoolinfo_neu.klassen");

CALL
sp_users("maximilian","1234dk","127.0.0.1","READ","schoolinfo_neu.klassen.beschreibung");
```

Code

sp_users.sql

```
DROP PROCEDURE IF EXISTS `sp_users`;

CREATE PROCEDURE `sp_users`(
  IN benutzer     VARCHAR(100),
  IN passwort     VARCHAR(100),
  IN hostname     VARCHAR(100),
  IN berechtigung VARCHAR(100),
  IN zugriffsort  VARCHAR(100)
)

  BEGIN
    /* Input Validation */
    IF benutzer IS NULL OR benutzer = ''
    THEN
      SIGNAL SQLSTATE 'ERROR'
      SET MESSAGE_TEXT = 'Ungültige Eingabe: Benutzer darf nicht leer sein!';
    END IF;
    IF passwort IS NULL OR passwort = ''
    THEN
      SIGNAL SQLSTATE 'ERROR'
      SET MESSAGE_TEXT = 'Ungültige Eingabe: Passwort darf nicht leer sein!';
    END IF;
    IF hostname IS NULL OR hostname = ''
    THEN
      SIGNAL SQLSTATE 'ERROR'
```

```
      SET MESSAGE_TEXT = 'Ungültige Eingabe: Hostname darf nicht leer sein!';
    END IF;
    IF berechtigung IS NULL OR berechtigung = ''
    THEN
      SIGNAL SQLSTATE 'ERROR'
      SET MESSAGE_TEXT = 'Ungültige Eingabe: Berechtigungsfeld darf nicht leer
sein!';
    END IF;

    INSERT INTO `schoolinfo_neu`.`log_berechtigung` (
      benutzer,
      timestamp,
      zugriffsort,
      typ,
      berechtigung,
      fuer
    ) VALUES (
      CURRENT_USER(),
      CURRENT_DATE(),
      hostname,
      zugriffsort,
      berechtigung,
      benutzer
    );
  END;
```

log_berechtigung.sql

```
USE schoolinfo_neu;

DROP TABLE IF EXISTS log_berechtigung;

CREATE TABLE IF NOT EXISTS log_berechtigung (
  id           INT
AUTO_INCREMENT,
  benutzer    VARCHAR(100)               NOT NULL,
  timestamp   DATETIME     NOT NULL
DEFAULT current_timestamp,
  wofuer VARCHAR(100) NOT NULL,
  typ         ENUM ('DB', 'TBL', 'ATTR') NOT NULL,
  berechtigung VARCHAR(100)              NOT NULL,
  fuer         VARCHAR(100)              NOT NULL,
  PRIMARY KEY (id)
);
```

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

sp_archives.sql

```
DROP PROCEDURE IF EXISTS `sp_archives`;

CREATE PROCEDURE `sp_archives`(
  IN klasse VARCHAR(255)
) BEGIN
  SET @id_klasse = (
    SELECT `id_klasse`
    FROM `schoolinfo_neu`.`klassen`
    WHERE `name` = klasse
    LIMIT 1
  );

  INSERT INTO `schoolinfo_archiv`.`lernende_archiv` (
    `anrede`,
    `name`,
    `vorname`,
    `geschlecht`,
    `bm`,
    `strasse`,
    `ort`,
    `plz`,
    `richtung`,
    `klasse`,
    `lehrbetrieb`,
    `lbstrasse`,
    `lbhausnr`,
    `lbplz`,
    `lbort`,
    `lbland`,
    `note_erf`,
    `note_knw`
  ) SELECT
      `lernender`.`anrede`,
      `lernender`.`name`,
      `lernender`.`vorname`,
```

```
      `lernender`.`geschlecht`,
      `lernender`.`bm`,
      `lernender`.`strasse`,
      `lernender_ort`.`name`,
      `lernender_ort`.`plz`,
      `klasse`.`name`,
      `klasse`.`beschreibung`,
      `richtung`.`name`,
      `lehrbetrieb`.`name`,
      `lehrbetrieb`.`strasse`,
      `lehrbetrieb`.`nummer`,
      `lehrbetrieb_ort`.`plz`,
      `lehrbetrieb_ort`.`name`,
      `note`.`note_erfahrung`,
      `note`.`note_knw`
    FROM `schoolinfo_neu`.`lernende` AS `lernender`
      LEFT JOIN `schoolinfo_neu`.`klassen` AS `klasse`
        ON `klasse`.`id_klasse` = `lernender`.`fk_klasse`
      LEFT JOIN `schoolinfo_neu`.`ortschaften` AS `lernender_ort`
        ON `lernender_ort`.`id_ort` = `lernender`.`fk_ort`
      LEFT JOIN `schoolinfo_neu`.`module` AS `richtung`
        ON `richtung`.`id_modul` = `lernender`.`fk_fachrichtung`
      LEFT JOIN `schoolinfo_neu`.`lehrbetriebe` AS `lehrbetrieb`
        ON `lehrbetrieb`.`id_lehrbetrieb` = `lernender`.`fk_lehrbetrieb`
      LEFT JOIN `schoolinfo_neu`.`ortschaften` AS `lehrbetrieb_ort`
        ON `lehrbetrieb_ort`.`id_ort` = `lehrbetrieb`.`fk_ort`
      LEFT JOIN `schoolinfo_neu`.`noten` AS `note`
        ON `note`.`fk_lernender` = `lernender`.`id_lernender`
    WHERE `id_klasse` = @id_klasse;

  DELETE FROM `schoolinfo_neu`.`noten`
  WHERE `fk_lernender` IN (
    SELECT *
    FROM (
          SELECT `lernende`.`id_lernender` AS `idl`
          FROM `schoolinfo_neu`.`noten` AS `note`
            LEFT JOIN `schoolinfo_neu`.`lernende` AS `lernender`
              ON `lernender`.`id_lernender` = `note`.`fk_lernender`
          WHERE `lernender`.`fk_klasse` = @id_klasse
        ) AS `auswahl`
  );

END;
```

schoolinfo_archiv.sql

```
CREATE DATABASE IF NOT EXISTS schoolinfo_archiv;

USE schoolinfo_archiv;

CREATE TABLE IF NOT EXISTS lernende_archiv (
  id          INT PRIMARY KEY       AUTO_INCREMENT,

  anrede      VARCHAR(25),
  name        VARCHAR(50)  NOT NULL,
  vorname     VARCHAR(50)  NOT NULL,
  geschlecht  ENUM ('M', 'F'),
  bm          TINYINT(1),
  strasse     VARCHAR(50),
  ort         VARCHAR(50),
  plz         VARCHAR(10),

  richtung    VARCHAR(50)  NOT NULL,

  klasse      VARCHAR(10)  NOT NULL,

  lehrbetrieb VARCHAR(100) NOT NULL,
  lbstrasse   VARCHAR(50),
  lbhausnr    VARCHAR(10),
  lbplz       VARCHAR(8),
  lbort       VARCHAR(50),
  lbland      VARCHAR(15),

  modulname   VARCHAR(50)  NOT NULL,

  note_erf    DECIMAL(3, 2),
  note_knw    DECIMAL(3, 2),

  timestamp   DATETIME     NOT NULL DEFAULT current_timestamp

);
```