KNW M-128 Datenbank Administrieren

IAP 12-16 B Jérôme Röthlisberger

Kompetenz-Nachweis Auftrag, Eine Datenbank der Gewerbeschule muss administriert werden. Dazu werden entsprechende Stored-Procedures entwickelt

---

Inhaltsverzeichnis

Projekt-Info

Autor	Jérôme Röthlisberger
E-Mail Adresse	jerome.roethlisberger@gibmit.ch
Projektmanager	T. Bögli, Lehrer der GIBM
Projekt E-Mail	t.boegli@gibmit.ch
Abgabe-Datum	

Dokument-Historie

Version	*Datum*	*Autor*	*Änderung*
1	*15.05.16*	*ROJE*	*Erstellt*
2	*16.05.16*	*ROJE*	*Bearbeitet*
3	*17.05.2016*	*ROJE*	*Abgabe*

Projekt Zusammenfassung

Ausgangslage

Die Applikation für Schuladministration der IT-Abteilung der Gewerblich Industriellen Berufsschule Muttenz ist soweit gewachsen dass eine Migration in eine neue Datenbank nötig ist.

Damit die Weiterentwicklung gewährleistet werden kann müssen Teile der Datenbank verändert werden.

Zu diesem Zweck werden entsprechende Stored-Procedures entwickelt mit denen einen Datenbank-Migration durchgeführt werden kann.

Umsetzung

Das Projekt ist in 4 Aufgabenbereiche gegliedert. Jedes dieser Aufgabenbereiche befasst sich mit einem Stored-Procedure.

Folgende Stored-Procedures werden umgesetzt

- sp_migration_starten
- sp_ortschaft_extrahieren
- sp_benutzer_erstellen
- sp_lernende_archivieren

1 Teilprojekt Datenbank Migration

Befasst sich mit Stored-Procedure: sp_migration_starten

1.1 Definition der Namenskonvention

1.1.1 Globale Konvention

- Datenbank Name
  - schoolinfo_neu
- Tabellennamen

    - Alphanumerisch
    - nur die Zeichen a-z und 0-9
    - Snake-Case sprich, Lehrzeichen durch Unterstriche ersetzt
    - kein Präfix
    - plural
    - deutsch
- Kolumnen
    - Alphanumerisch
    - nur die Zeichen a-z und 0-9
    - Snake-Case sprich, Lehrzeichen durch Unterstriche ersetzt
    - kein Präfix
    - singular
    - deutsch
- Tabellenaufbau
    - IMMER nur eine Kolumne ist Primärschlüssel
    - Primärschlüssel-Kolumne ist immer
        - Name: id
        - Datentyp: INTEGER(10) NOT NULL AUTOINCREMENT
    - Fremdschlüssel Kolumne sind immer
        - Konvention: Kolumn name ist IMMER singular.
        - Name: fremdtabelle_id
          fremndtabelle_kontext_id
        - Datentyp: INTEGER(10)
    - Fremdschlüssel
        - Konvention: Fremdschlüssel sind IMMER singular und zeigen IMMER auf eine id.
        - Name eigentabelle_fremdtabelle_id
          eigentabelle_fremdtabelle_kontext_id

1.1.2 Zusätzliche Definitionen

Kolumnen die Text enthalten werden grundsätzlich auf VARCHAR(50) gesetzt. Ausnahmen entstehen bei Felder die in der Quelldatenbank mit mehr als 50 Zeichen definiert wurden. Diese werden grundsätzlich auf VARCHAR(255) gesetzt.

Postleitzahlen werden auf VARCHAR(8) gesetzt um somit den deutschen Postleitzahlen Rechnung zu tragen.

Situativ können auch VARCHAR(2) verwendet werden.

1.1.3 Datenkatalog
Datenbank: schoolinfo_neu

1.1.4 Tabelle Klassen
Tabelle Klassen

id	INT(10)	NOT NULL	AUTO INCREMENT
lehrer_id	INT(10)		DEFAULT NULL
name	VARCHAR(50)	NOT NULL	
beschreibung	VARCHAR(255)		DEFAULT NULL

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			id
FKY	klassen_lehrer_id	lehrer	id	lehrer_id

1.1.5 Tabelle Lehrbetriebe
Tabelle lehrbetriebe

id	INT(10)	NOT NULL	AUTO INCREMENT
name	VARCHAR(50)		DEFAULT NULL
strasse	VARCHAR(50)	NOT NULL	
haus_nr	VARCHAR(50)		DEFAULT NULL
plz	VARCHAR(8)	NOT NULL	
ort	VARCHAR(50)	NOT NULL	
kanton_code	VARCHAR(2)		DEFAULT NULL
land_code	VARCHAR(2)		DEFAULT NULL

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			id

1.1.6 Tabelle Lernende

Tabelle lernende			
id	INT(10)	NOT NULL	AUTO INCREMENT
anrede	VARCHAR(50)		DEFAULT NULL
name	VARCHAR(50)	NOT NULL	
vorname	VARCHAR(50)	NOT NULL	
geschlecht	VARCHAR(2)	NOT NULL	“M“ = Männlich, “F“ = Weiblich
klasse_id	INT(10)	NOT NULL	
ist_bm	TINYINT(1)	NOT NULL	0 = Nein 1 = Ja
fachrichtung_id	INT(10)	NOT NULL	
lehrbetrieb_id	INT(10)	NOT NULL	
strasse	VARCHAR(50)	NOT NULL	
plz	VARCHAR(8)	NOT NULL	
ort	VARCHAR(50)	NOT NULL	

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			
FKY	lernende_fachrichtung_id	fachrichtungen	id	fachrichtung_id
FKY	lernende_klasse_id	klassen	id	klasse_id
FKY	lernende_lehrbetrieb_id	lehrbetriebe	id	lehrbetrieb_id

1.1.7 Tabelle Berechtigungen_log

Tabelle berechtigungen_log			
id	INT(10)	NOT NULL	AUTO INCREMENT
benutzer	VARCHAR(50)	NOT NULL	
zeitpunkt	DATETIME	NOT NULL	DEFAULT CURRENT_TIMESTAMP
grund	VARCHAR(50)	NOT NULL	
berechtigung	VARCHAR(50)	NOT NULL	
tabelle	VARCHAR(50)	NOT NULL	

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			id

1.1.8 Tabelle module

Tabelle module

id	INT(10)	NOT NULL	AUTO INCREMENT
benutzer	VARCHAR(50)	NOT NULL	
zeitpunkt	DATETIME	NOT NULL	DEFAULT CURRENT_TIMESTAMP
grund	VARCHAR(50)	NOT NULL	
berechtigung	VARCHAR(50)	NOT NULL	
tabelle	VARCHAR(50)	NOT NULL	

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			id

1.1.9 Tabelle noten

Tabelle noten

id	INT(10)	NOT NULL	AUTO INCREMENT
lernende_id	INT(10)	NOT NULL	
modul_id	INT(10)	NOT NULL	
erf_note	DOUBLE(15,2)	NOT NULL	
erf_datum	DATETIME	NOT NULL	
knw_note	DOUBLE(15,2)	NOT NULL	
knw_datum	DATETIME	NOT NULL	

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			id
FKY	noten_lernende_id	lernende	id	lernende_id
FKY	noten_modul_id	module	id	modul_id

1.1.10 Tabelle Fachrichtungen

Tabelle fachrichtungen

id	INT(10)	NOT NULL	AUTO INCREMENT
name	VARCHAR(50)	NOT NULL	

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			id

1.2 Stored Procedure sp_migration_starten

- Name
  - sp_migration_starten
- Eingabeparameter
  - keine
- Ausgabeparameter
  - keine
- Resultat
  - Datenbank schoolinfo_neu ist erstellt
  - Folgende Tabellen sind erstellt
    - klassen
    - lehrbetriebe
    - lernende
    - berechtigunen_log
    - module
    - noten
    - fachrichtungen
  - Daten aus Datenbank schoolinfo12802016 sind in die Tabellen von schoolinfo_neu importiert.

1.3 Stored Procedure Quelltext

```
DELIMITER //
CREATE PROCEDURE sp_migration_starten
  BEGIN
    DROP DATABASE IF EXISTS `schoolinfo_neu`;
    CREATE DATABASE IF NOT EXISTS `schoolinfo_neu`
      DEFAULT CHARACTER SET latin1;
    USE `schoolinfo_neu`;
    SET SESSION SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";

    START TRANSACTION;
    DROP TABLE IF EXISTS `klassen`;
    CREATE TABLE `klassen` (
      `id`           INT(10) NOT NULL        AUTO_INCREMENT,
      `lehrer_id`    INT(10)                 DEFAULT NULL,
      `name`         VARCHAR(50)
                                             DEFAULT NULL,
      `beschreibung` VARCHAR(255)
                                             DEFAULT NULL,
      PRIMARY KEY (`id`)
      #   FOREIGN KEY (klassen_lehrer_id) REFERENCES lehrer(id)
    )
      ENGINE = InnoDB
      DEFAULT CHARSET = utf8
      COLLATE = utf8_unicode_ci;
    SELECT "Table klassen has been created";
    COMMIT;

    START TRANSACTION;
    LOCK TABLES `klassen` WRITE;
```

```
ALTER TABLE `klassen`
  DISABLE KEYS;
INSERT INTO `klassen` (`id`, `lehrer_id`, `name`, `beschreibung`)
  SELECT
    `idklasse`      AS `id`,
    `klassenlehrer` AS `lehrer_id`,
    `name`          AS `name`,
    `realname`      AS `beschreibung`
  FROM `schoolinfo12802016`.`klasse`;
ALTER TABLE `klassen`
  ENABLE KEYS;
UNLOCK TABLES;
SELECT "Data for klassen has been imported";
COMMIT;

START TRANSACTION;
DROP TABLE IF EXISTS `lehrbetriebe`;
CREATE TABLE `lehrbetriebe` (
  `id`          INT(10)     NOT NULL        AUTO_INCREMENT,
  `name`        VARCHAR(50) NOT NULL,
  `strasse`     VARCHAR(50) NOT NULL,
  `haus_nr`     VARCHAR(8)                  DEFAULT NULL,
  `plz`         VARCHAR(8)  NOT NULL,
  `ort`         VARCHAR(50) NOT NULL,
  `kanton_code` VARCHAR(2)                  DEFAULT NULL,
  `land_code`   VARCHAR(2)                  DEFAULT NULL,
  PRIMARY KEY (`id`)
)
  ENGINE = InnoDB
  DEFAULT CHARSET = utf8
  COLLATE = utf8_unicode_ci;
SELECT "Table lehrbetriebe has been created";
COMMIT;

START TRANSACTION;
LOCK TABLES `lehrbetriebe` WRITE;
ALTER TABLE `lehrbetriebe`
  DISABLE KEYS;
INSERT INTO `lehrbetriebe` (
  `id`, `name`, `strasse`, `haus_nr`, `plz`, `ort`, `kanton_code`, `land_code`
)
  SELECT
    `id_Lehrbetrieb` AS `id`,
    `FName`          AS `name`,
    `FStrasse`       AS `strasse`,
    `FHausNr`        AS `haus_nr`,
    `FPlz`           AS `plz`,
    `FOrt`           AS `ort`,
    `FKanton`        AS `kanton_code`,
    `FLand`          AS `land_code`
```

```
  FROM `schoolinfo12802016`.`lehrbetriebe`;

ALTER TABLE `lehrbetriebe`
  ENABLE KEYS;
UNLOCK TABLES;
SELECT "Data for lehrbetriebe has been imported";
COMMIT;

START TRANSACTION;
DROP TABLE IF EXISTS `lernende`;
CREATE TABLE `lernende` (
  `id`              INT(10)       NOT NULL     AUTO_INCREMENT,
  `anrede`          VARCHAR(25)                DEFAULT NULL,
  `name`            VARCHAR(50)                DEFAULT NULL,
  `vorname`         VARCHAR(50)                DEFAULT NULL,
  `geschlecht`      VARCHAR(50)                DEFAULT NULL,
  `klasse_id`       INT(10)                    DEFAULT NULL,
  `ist_bm`          TINYINT(1)    NOT NULL,
  `fachrichtung_id` INT(10)                    DEFAULT NULL,
  `lehrbetrieb_id`  INT(10)                    DEFAULT NULL,
  `strasse`         VARCHAR(50)                DEFAULT NULL,
  `plz`             VARCHAR(50)                DEFAULT NULL,
  `ort`             VARCHAR(50)                DEFAULT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (lernende_klasse_id) REFERENCES klassen (id),
  FOREIGN KEY (lernende_fachrichtung_id) REFERENCES fachrichtungen (id),
  FOREIGN KEY (lernende_lehrbetrieb_id) REFERENCES lehrbetriebe (id)
)
  ENGINE = InnoDB
  DEFAULT CHARSET = utf8
  COLLATE = utf8_unicode_ci;
SELECT "Table lernende has been created";
COMMIT;

START TRANSACTION;
LOCK TABLES `lernende` WRITE;
ALTER TABLE `lernende`
  DISABLE KEYS;
INSERT INTO `lernende` (
  `id`, `anrede`, `name`, `vorname`,
  `geschlecht`, `klasse_id`, `ist_bm`,
  `fachrichtung_id`, `lehrbetrieb_id`,
  `strasse`, `plz`, `ort`
)
  SELECT
    `Lern_id`     AS `id`,
    `anrede`      AS `anrede`,
    `name`        AS `name`,
    `vorname`     AS `vorname`,
    `geschlecht`  AS `geschlecht`,
    `klasse`      AS `klasse_id`,
```

```
    `bm`          AS `ist_bm`,
    `richtung`    AS `fachrichtung_id`,
    `lehrbetrieb` AS `lehrbetrieb_id`,
    `strasse`     AS `strasse`,
    `plz`         AS `plz`,
    `ort`         AS `ort`
  FROM `schoolinfo12802016`.`lernende`;

ALTER TABLE `lernende`
  ENABLE KEYS;
UNLOCK TABLES;
SELECT "Data for lernende has been imported";
COMMIT;

START TRANSACTION;
DROP TABLE IF EXISTS `berechtigungen_log`;
CREATE TABLE `berechtigungen_log` (
  `id`           INT(10)      NOT NULL AUTO_INCREMENT,
  `benutzer`     VARCHAR(50)  NOT NULL,
  `zeitpunkt`    DATETIME     NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `grund`        VARCHAR(50)  NOT NULL,
  `berechtigung` VARCHAR(50)  NOT NULL,
  `tabelle`      VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`)
)
  ENGINE = InnoDB
  DEFAULT CHARSET = latin1;
SELECT "Table berechtigungen_log has been created";
COMMIT;

START TRANSACTION;
LOCK TABLES `berechtigungen_log` WRITE;
ALTER TABLE `berechtigungen_log`
  DISABLE KEYS;
INSERT INTO `berechtigungen_log` (
  `id`, `benutzer`, `zeitpunkt`, `grund`, `berechtigung`, `tabelle`)
  SELECT
    `id`           AS `id`,
    `benutzer`     AS `benutzer`,
    `timestamp`    AS `zeitpunkt`,
    `wofuer`       AS `grund`,
    `berechtigung` AS `berechtigung`,
    `fuer`         AS `tabelle`
  FROM `schoolinfo12802016`.`log_berechtigung`;
ALTER TABLE `berechtigungen_log`
  ENABLE KEYS;
UNLOCK TABLES;
SELECT "Data for berechtigungen_log has been imported";
COMMIT;

START TRANSACTION;
```

```sql
DROP TABLE IF EXISTS `module`;
CREATE TABLE `module` (
  `id`           INT(10) NOT NULL        AUTO_INCREMENT,
  `name`         VARCHAR(50)             DEFAULT NULL,
  `beschreibung` VARCHAR(255)            DEFAULT NULL,
  PRIMARY KEY (`id`)
)
  ENGINE = InnoDB
  DEFAULT CHARSET = utf8
  COLLATE = utf8_unicode_ci;
SELECT "Table module has been created";
COMMIT;

START TRANSACTION;
LOCK TABLES `module` WRITE;
ALTER TABLE `module`
  DISABLE KEYS;
INSERT INTO `module` (
  `id`, `name`, `beschreibung`
)
  SELECT
    `id`        AS `id`,
    `m_name`    AS `name`,
    `modulname` AS `beschreibung`
  FROM `schoolinfo12802016`.`modul`;


ALTER TABLE `module`
  ENABLE KEYS;
UNLOCK TABLES;
SELECT "Data for module has been imported";
COMMIT;

START TRANSACTION;
DROP TABLE IF EXISTS `noten`;
CREATE TABLE `noten` (
  `id`          INT(10) NOT NULL AUTO_INCREMENT,
  `lernende_id` INT(10)          DEFAULT NULL,
  `modul_id`    INT(10)          DEFAULT NULL,
  `erf_note`    DOUBLE(15, 2)    DEFAULT NULL,
  `erf_datum`   DATETIME         DEFAULT NULL,
  `knw_note`    DOUBLE(15, 2)    DEFAULT NULL,
  `knw_datum`   DATETIME         DEFAULT NULL,
  PRIMARY KEY (`id`)
)
  ENGINE = InnoDB
  DEFAULT CHARSET = utf8
  COLLATE = utf8_unicode_ci;
SELECT "Table noten has been created";
COMMIT;
```

```
START TRANSACTION;
LOCK TABLES `noten` WRITE;
ALTER TABLE `noten`
  DISABLE KEYS;

INSERT INTO `noten` (
  `lernende_id`,
  `modul_id`,
  `erf_note`,
  `erf_datum`,
  `knw_note`,
  `knw_datum`
)
  SELECT
    `lernende_idlernende` AS `lernende_id`,
    `module_idmodule`     AS `modul_id`,
    `erf_note`            AS `erfahrungsnote`,
    `erf_datum`           AS `erfahrungsnote_datum`,
    `knw_note`            AS `knw_note`,
    `knw_datum`           AS `knw_datum`
  FROM `schoolinfo12802016`.`noten`;

ALTER TABLE `noten`
  ENABLE KEYS;
UNLOCK TABLES;
SELECT "Data noten has been imported";
COMMIT;

START TRANSACTION;
DROP TABLE IF EXISTS `fachrichtungen`;
CREATE TABLE `fachrichtungen` (
  `id`   INT(10) NOT NULL        AUTO_INCREMENT,
  `name` VARCHAR(50)             DEFAULT NULL,
  PRIMARY KEY (`id`)
)
  ENGINE = InnoDB
  DEFAULT CHARSET = utf8
  COLLATE = utf8_unicode_ci;
SELECT "Table fachrichtungen has been created";
COMMIT;

START TRANSACTION;
LOCK TABLES `fachrichtungen` WRITE;
ALTER TABLE `fachrichtungen`
  DISABLE KEYS;

INSERT INTO `fachrichtungen` (
  `id`, `name`
)
  SELECT
    `id`       AS `idrichtung`,
```

```
        `richtung` AS `name`
      FROM `schoolinfo12802016`.`richtung`;

    ALTER TABLE fachrichtungen
      ENABLE KEYS;
    UNLOCK TABLES;
    SELECT "Data fachrichtungen has been imported";
    COMMIT;
  END //
DELIMITER ;
```

2 Teilprojekt Ortschaftstbelle erstellen

2.1 Stored Procedure sp_orte_extrahiern

- Name
  - sp_orte_extrahieren
- Eingabeparameter
  - keine
- Ausgabeparameter
  - keine
- Resultat
  - Tabelle schoolinfo_neu.orte ist erstellt
  - Daten sind aus folgenden Tabellen zusammengeführt
    - schoolinfo_neu.lernende
    - schoolinfo_neu.lehrbetriebe
  - Folgende Tabellen sind verändert
    - schoolinfo_neu.lernende
    - schoolinfo_neu.lehrbetriebe

2.1.1 Datenkatalog

2.1.2 Tabelle Orte

Tabelle orte			
id	INT(10)	NOT NULL	AUTO INCREMENT
plz	VARCHAR(8)	NOT NULL	
ort	VARCHAR(50)		

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			id

2.1.3 Tabelle Lehrbetriebe

Tabelle lehrbetriebe			
id	INT(10)	NOT NULL	AUTO INCREMENT
name	VARCHAR(50)		DEFAULT NULL
strasse	VARCHAR(50)	NOT NULL	
haus_nr	VARCHAR(50)		DEFAULT NULL
ort_id	INT(10)	NOT NULL	
kanton_code	VARCHAR(2)		DEFAULT NULL
land_code	VARCHAR(2)		DEFAULT NULL

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			id
FKY	lehrbetriebe_ort_id	orte	id	ort_id

2.1.4 Tabelle Lernende

Tabelle lernende			
id	INT(10)	NOT NULL	AUTO INCREMENT
anrede	VARCHAR(50)		DEFAULT NULL
name	VARCHAR(50)	NOT NULL	
vorname	VARCHAR(50)	NOT NULL	
geschlecht	VARCHAR(2)	NOT NULL	“M“ = Männlich, “F“ = Weiblich
klasse_id	INT(10)	NOT NULL	
ist_bm	TINYINT(1)	NOT NULL	0 = Nein 1 = Ja
fachrichtung_id	INT(10)	NOT NULL	
lehrbetrieb_id	INT(10)	NOT NULL	
strasse	VARCHAR(50)	NOT NULL	
ort_id	INT(10)	NOT NULL	

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			
FKY	lernende_fachrichtung_id	fachrichtungen	id	fachrichtung_id
FKY	lernende_klasse_id	klassen	id	klasse_id
FKY	lernende_lehrbetrieb_id	lehrbetriebe	id	lehrbetrieb_id
FKY	lernende_ort_id	orte	id	ort_id

2.2 Stored Procedure Quelltext

```
DELIMITER //
CREATE PROCEDURE sp_orte_extrahieren
  BEGIN
  START TRANSACTION;
    DROP TABLE IF EXISTS `orte`;
    CREATE TABLE `orte` (
      `id`    INT(10)            NOT NULL        AUTO_INCREMENT,
      `plz`   VARCHAR(6)         NOT NULL,
      `ort` VARCHAR(50)          NOT NULL,
      `region_code` VARCHAR(2)   DEFAULT NULL,
      'land_code'   VARCHAR(2)   DEFAULT NULL,
      PRIMARY KEY (`id`)
    )
      ENGINE = InnoDB
      DEFAULT CHARSET = utf8
      COLLATE = utf8_unicode_ci;
    SELECT "Table ort has been created";
    COMMIT;

    START TRANSACTION;
    LOCK TABLES `orte` WRITE;
    ALTER TABLE `orte` DISABLE KEYS;
    INSERT INTO `orte` (`plz`, `ort`)

      SELECT DISTINCT
        plz, ort
      FROM (
             SELECT
               TRIM(TRAILING "\r" FROM `plz`
               ) as plz,
               TRIM(TRAILING " 1" FROM
                    TRIM(TRAILING " AG" FROM
                         TRIM(TRAILING " SO" FROM
                              TRIM(TRAILING " BL" FROM
                                   TRIM(TRAILING "\r" FROM `ort`)
                              )
                         )
                    )
               ) as ort
             FROM `lernende` GROUP BY `plz`, `ort`
             UNION
             SELECT
               TRIM(TRAILING "\r" FROM `plz`
               ) as plz,
               TRIM(TRAILING " 1" FROM
                    TRIM(TRAILING " AG" FROM
                         TRIM(TRAILING " SO" FROM
                              TRIM(TRAILING " BL" FROM
```

```
                                TRIM(TRAILING "\r" FROM `ort`)
                            )
                        )
                    )
                ) as ort
            FROM `lehrbetriebe` GROUP BY `plz`, `ort`) as tmp;

ALTER TABLE `orte` ENABLE KEYS;
UNLOCK TABLES;
SELECT "Data for orte has been imported";
COMMIT;

START TRANSACTION;
  LOCK TABLES `lernende` WRITE;
  ALTER TABLE `lernende`
    ADD COLUMN `ort_id` INT(10) NULL DEFAULT NULL AFTER `strasse`;
  SELECT "ort collumn has been created for lernende table";

  UPDATE `lernende`
    SET `lernende`.`ort_id` = `orte`.`id`
    WHERE
    `lernende`.`plz` LIKE CONCAT('%', `orte`.`plz` ,'%') AND
    `lernende`.`ort` LIKE CONCAT('%', `orte`.`ort` ,'%');
  SELECT "ort_id has been set in lernende table";

  ALTER TABLE `lernende`
      DROP COLUMN `plz`,
    DROP COLUMN `ort`,
    ADD FOREIGN KEY (`lernende_ort_id`)
    REFERENCES `orte`(`id`);

  SELECT "lernende_ort_id foreign key has been established";
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
  LOCK TABLES `lehrbetriebe` WRITE;
  ALTER TABLE `lehrbetriebe`
    ADD COLUMN `ort_id` INT(10) NULL DEFAULT NULL AFTER `haus_nr`;
  SELECT "ort collumn has been created for lehrbetriebe table";

  UPDATE `lehrbetriebe`
    SET `lehrbetriebe`.`ort_id` = `orte`.`id`
    WHERE
    `lehrbetriebe`.`plz` LIKE CONCAT('%', `orte`.`plz` ,'%') AND
    `lehrbetriebe`.`ort` LIKE CONCAT('%', `orte`.`ort` ,'%');
  SELECT "ort_id has been set in lehrbetriebe table";

  ALTER TABLE `lehrbetriebe`
    DROP COLUMN `plz`,
    DROP COLUMN `ort`,
```

```
        ADD FOREIGN KEY (`lernende_ort_id`)
        REFERENCES `orte`(`id`);

      SELECT "lehrbetriebe_ort_id foreign key has been established";
      UNLOCK TABLES;
    COMMIT;
  END //
DELIMITER ;
```

3 Teilprojekt Benutzer erstellen

3.1 Stored Procedure sp_benutzer_erstellen

- Name
  - sp_benutzer_erstellen
- Eingabeparameter
  - IN this_user VARCHAR(50)
  - IN this_host VARCHAR(50)
  - IN this_pass VARCHAR(50)
  - IN this_table VARCHAR(50)
  - IN this_permission VARCHAR(50)
- Ausgabeparameter
  - keine
- Resultat
  - Benutzer ist erstellt.
  - Benutzer hat die Berechtigung für die entsprechende Tabelle gesetzt
  - Logtabelle erstellt sofern diese nicht bereits existiert.
  - Logeintrag in Tabelle schoolinfo_neu.berechtigungen_log erstellt
  - Bei Fehleingaben wird eine Exception geworfen

3.2 Aufrufbeispiele

Benutzer erstellen und Leseberechtigung für Tabelle schoolinfo_neu.lernende erteilen

```
sp_bentuzer_erstellen("jerome","localhost","1234","lernende","SELECT");
```

3.3 Eingabe Validation

Die Eingabeparameter werden validiert. Die maximale Zeichenlänge von 50 Zeichen pro Parameter wird berücksichtigt. Ausserdem wird nach potentiell schädlichen Zeichen geprüft. Diese Prüfung erfolgt mittels REGEX

Siehe Persönliches Lernjournal

http://wiki.steampilot.ch → Suche → Regex

3.4 Datenkatalog

3.4.1 Tabelle Berechtigungen_log

Tabelle berechtigungen_log

id	INT(10)	NOT NULL	AUTO INCREMENT
benutzer	VARCHAR(50)	NOT NULL	
zeitpunkt	DATETIME	NOT NULL	DEFAULT CURRENT_TIMESTAMP
grund	VARCHAR(50)	NOT NULL	
berechtigung	VARCHAR(50)	NOT NULL	
tabelle	VARCHAR(50)	NOT NULL	

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			id

3.5 Stored Procedure Quelltext

```
DELIMITER //
CREATE PROCEDURE sp_benutzer_erstellen(
  IN this_user       VARCHAR(50),
  IN this_host       VARCHAR(50),
  IN this_pass       VARCHAR(50),
  IN this_table      VARCHAR(50),
  IN this_permission VARCHAR(50)
)
  BEGIN
    START TRANSACTION;
    IF (LENGTH(
      this_user OR
      this_host OR
      this_pass OR
      this_table OR
      this_permission
       ) > 50)
      THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Eingabe zu lange';
    ELSEIF (
      this_user OR
      this_host OR
      this_pass OR
      this_table OR
      this_permission
    REGEXP  '|^((?![;$<>{}\[\]]).)*$|')
      THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Eingabe enthaelt unerlaubte Zeichen. Nur a-z A-Z 0-9 sind
                            erlaubt';
    END IF;

    DECLARE query1 VARCHAR(255);
    SET query1 = CONCAT('
       CREATE USER IF NOT EXISTS "', this_user, '"@"', this_host,
                       '" IDENTIFIED BY "', this_pass, '" ');

    PREPARE statement1 FROM query1;
    EXECUTE statement1;
    DEALLOCATE PREPARE statement1;

    SELECT CONCAT(' User ', this_user, ' at ', this_host, ' has been created.');

    INSERT INTO `schoolinfo_neu`.`berechtigungen_log`
    (`benutzer`, `grund`, `berechtigung`,`tabelle`)
    VALUES (CONCAT(this_user, "@", this_host), "CREATE", "NONE", "NONE");
    COMMIT;
```

```
    START TRANSACTION;
    DECLARE query2 VARCHAR(255);
    SET query2 = CONCAT(
        'GRANT ', this_permission, ' ON `schoolinfo.neu`.', this_table,
        ' TO "', this_user, '"@"', this_host, '" IDENTIFIED BY "', this_pass, '" ');
    PREPARE statement2 FROM query2;
    EXECUTE statement2;
    DEALLOCATE PREPARE statement2;
    SELECT CONCAT(' User ', this_user, ' at ', this_host,
                  ' has been granted ',this_permission,
                  ' permisson on table ',this_table,'.');
    INSERT INTO `schoolinfo_neu`.`berechtigungen_log`
    (`benutzer`, `grund`, `berechtigung`, `tabelle`)
    VALUES (CONCAT(this_user, "@", this_host), "Granted permission", this_permission,
this_table);
    COMMIT;

  END //
DELIMITER ;
```

4 Teilprojekt Lernende archivieren

4.1 Stored Procedure sp_lernende_archivieren

- Name
  - sp_lernende_archivieren
- Eingabeparameter
  - IN this_klasse_id INT(10)
- Ausgabeparameter
  - keine
- Resultat
  - Datenbank schoolinfo_archiev ist erstellt sofern diese nicht bereits existiert
  - Tabelle schoolinfo_archiv.lernende_archiv ist erstellt sofern diese nicht bereits existiert
  - Archieveinträge aus den zusammengezogenen Daten aus Datenbank schoolinfo_neu ist in Tabelle lernende_archiv geschrieben

4.2 Datenkatalog

4.2.1 Tabelle schoolinfo_archiv.lernende_archiv

Tabelle lernende_archiv

id	INT(10)	NOT NULL	AUTO INCREMENT
nachname	VARCHAR(50)	NOT NULL	
vorname	VARCHAR(80)	NOT NULL	
bm	BOOLEAN	NOT NULL	
strasse	VARCHAR(50)	NOT NULL	
plz	VARCHAR(8)	NOT NULL	
ort	VARCHAR(50)	NOT NULL	
richtung	VARCHAR(50)	NOT NULL	
klasse	VARCHAR(50)	NOT NULL	
lehrbetrieb	VARCHAR(50)	NOT NULL	
lbstrasse	VARCHAR(50)	NOT NULL	
lbhausnr	VARCHAR(8)	NOT NULL	
lbplz	VARCHAR(8)	NOT NULL	
lbort	VARCHAR(50)	NOT NULL	
lbland	VARCHAR(2)	NOT NULL	
modulname	VARCHAR(50)	NOT NULL	
note_erf	DOUBLE(15,2)	NOT NULL	
note_knw	DOUBLE(15,2)	NOT NULL	

SCHLÜSSEL	***Name***	***Referenztabelle***	***Feld***	***Referenz Feld***
PKY	id			id

4.3 Stored Procedure Quelltext

```
DELIMITER //
CREATE PROCEDURE sp_lernende_archivieren(
  IN this_klasse_id INT(10)
)
  BEGIN
    CREATE DATABASE IF NOT EXISTS schoolinfo_archiv;

    USE schoolinfo_archiv;

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

      nachname    VARCHAR(50)  NOT NULL,
      vorname     VARCHAR(50)  NOT NULL,
      bm          TINYINT(1)   NOT NULL,
      strasse     VARCHAR(50)  NOT NULL,
      plz         VARCHAR(8)   NOT NULL,
      ort         VARCHAR(50)  NOT NULL,

      richtung    VARCHAR(30)  NOT NULL,

      klasse      VARCHAR(10)  NOT NULL,

      lehrbetrieb VARCHAR(100) NOT NULL,
      lbstrasse   VARCHAR(50)  NOT NULL,
      lbhausnr    VARCHAR(8)   NOT NULL,
      lbplz       VARCHAR(8)   NOT NULL,
      lbort       VARCHAR(50)  NOT NULL,
      lbland      VARCHAR(2)   NOT NULL,

      modulname   VARCHAR(50)  NOT NULL,

      note_erf    DOUBLE(15, 2)          DEFAULT NULL,
      note_knw    DOUBLE(15, 2)          DEFAULT NULL,

      timestamp   DATETIME     NOT NULL DEFAULT current_timestamp

    )
      ENGINE = ARCHIVE;

    INSERT INTO schoolinfo_archiv.lernende_archiv
      SELECT
        LD.`name`         AS nachname,
        LD.`vorname`      AS vorname,
        LD.`ist_bm`       AS bm,
        LD.`strasse`      AS strasse,
        LT.`plz`          AS plz,
        LT.`ort`          AS ort,
```

```
        LD.`fachrichtung` AS richtung,
        KL.`name`         AS klasse,
        BB.`name`         AS lehrbetrieb,
        BB.`strasse`      AS lbstrasse,
        BB.`haus_nr`      AS lbhausnr,
        BT.`plz`          AS lbplz,
        BT.`ort`          AS lbort,
        BB.`land_code`    AS lbland,
        MD.`name`         AS modulname,
        NT.`erf_note`     AS note_erf,
        NT.`knw_note`     AS note_knw
      FROM schoolinfo_neu.lernende AS LD
        LEFT JOIN schoolinfo_neu.orte AS LT
          ON LD.ort_id = LT.id
        LEFT JOIN schoolinfo_neu.klasse AS KL
          ON LD.klasse_id = KL.id
        LEFT JOIN schoolinfo_neu.lehrbetriebe AS BB
          ON LD.lehrbetrieb_id = BB.id
        LEFT JOIN schoolinfo_neu.orte AS BT
          ON BB.ort_id = BT.id
        RIGHT OUTER JOIN schoolinfo_neu.note AS NT
          ON NT.lernende_id = LD.id
        LEFT JOIN schoolinfo_neu.module AS MD
          ON NT.modul_id = MD.id
      WHERE LD.klasse_id = this_klasse_id;

    DELETE t1 FROM schoolinfo_neu.noten t1
      INNER JOIN schoolinfo_neu.lernende t2 ON (t1.lernende_id = t2.id)
      INNER JOIN schoolinfo_neu.klasse t3 ON (t2.klasse_id = t3.id)
    WHERE t3.id = this_klasse_id;

    DELETE FROM schoolinfo_neu.lernende AS LD
      WHERE LD.klasse_id = this_klasse_id;

  END //
DELIMITER ;
```