# Synchronisation von heterogenen Datenbanken
# SYT - 5A HIT

Ahmed Aly, Martin Haidn

November 28, 2014

## Contents

# 1 Aufgabenstellung

Dokumentieren Sie Ihren Versuch zwei heterogene Datenbanksysteme (MySQL, Postgresql) zu synchronisieren. Verwenden Sie dabei unterschiedliche Schemata (verschiedene Tabellenstruktur) und zeigen Sie auf, welche Schwierigkeiten bei den unterschiedlichen Heterogenitätsgraden auftreten können (wie im Unterricht besprochen) [2Pkt].

Implementieren Sie eigenständig eine geeignete Middleware [8Pkt]. Testen Sie Ihr gewähltes System mit mehr als einer Tabelle 4Pkt und dokumentieren Sie die Funktionsweise, sowie auch die Problematiken bzw. nicht abgedeckte Fälle [2Pkt].

Das PDF soll ausführlich beschreiben, welche Annahmen getroffen wurden. Der Source-Code muss den allgemeinen Richtlinien entsprechen und ebenfalls abgegeben werden.

Gruppengröße: 2 Gesamtpunkte: 16 [Aufteilung in eckigen Klammern ersichtlich]

**Punkte**

Dokumentation der Synchronisation [2Pkt]

Implementierung der Middleware [8Pkt]

- Zeittrigger bzw. Listener für Synchronisation bzw. DBMS Logs
- Konfiguration bez. Mapping der Tabellen und Attribute
- Konfliktlösung bei Zeitüberschneidung bzw. Datenproblemen (Log)
- LostUpdate-Problem

Test mit mehr als einer Tabelle und mindestens 10 Datensätze pro Tabelle [4Pkt]

- Uni- und Bidirektionale Änderungen mehrerer Tabellen
- Einfügen, Ändern und Löschen

Dokumenation der Funktionsweise, Problematiken und Problemfälle [2Pkt]

- Designdokumentation (Code + DB)
- Synchronisationsverhalten
- unbehandelte Problemfälle

Protokoll und Sourcecodedokumentation [0..-6Pkt]

# 2 Aufwandsschätzung

## 2.1 Schätzung im Vorfeld

Für die Durchführung dieser Übung sind elf Stunden vorgesehen. Die Aufgabenstellung wurde in Arbeitspakete untergliedert und für die jeweiligen Tasks wurde eine Zeitaufwandsschätzung erstellt.

| Arbeitszeitaufzeichnung - DBSync | | | |
|---|---|---|---|
| **Task:** | **Mitglied:** | **Geschätzte Zeit:** | **Benötigte Zeit:** |
| Vorbereitung & Evaluierung | | 1.5 | |
| Datenbankentwurf (Bleistift ERD u. RM) | | 0.75 | |
| Create-Script | | 0.5 | |
| Middleware Konzept | | 1 | |
| Realisierung der Middleware | | 5 | |
| Testing | | 1 | |
| Dokumentaion & Organisation | | 1.5 | |
| | **Gesamt:** | 11.25 | |

| | |
|---|---|
| Task... | Durchzuführendes Arbeitspaket |
| Mitglied... | Bearbeitendes Teammitglied |
| Geschätze Zeit... | Geschätzte Zeit für den jeweiligen task in (Stunden) |
| Benötigte Zeit... | Tatsächlich investierte Zeit in den jeweiligen Task (Stunden) |

## 2.2 Tatsächlich benötigt

Wie sich in der tatsächlich benötigten Zeit ablesen lässt, ist bei der Vorbereitung schon um einiges mehr Zeit vergangen als erwartet. Grund dafür war das ewige trouble shooting bezüglich der Propel installation und pear.

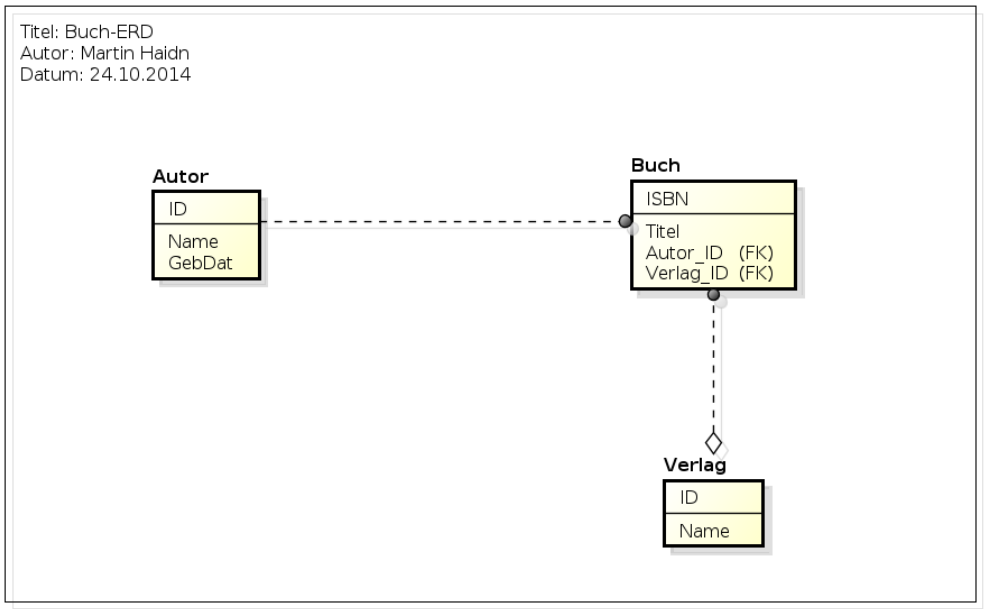
| Arbeitszeitaufzeichnung - DBSync | | | |
|---|---|---|---|
| **Task:** | **Mitglied:** | **Geschätzte Zeit:** | **Benötigte Zeit:** |
| Vorbereitung & Evaluierung | | 1.5 | 4 |
| Datenbankentwurf (Bleistift ERD u. RM) | | 0.75 | 2 |
| Create-Script | | 0.5 | 0.5 |
| Middleware Konzept | | 1 | 0.5 |
| Realisierung der Middleware | | 5 | 4 |
| Testing | | 1 | |
| Dokumentaion & Organisation | | 1.5 | 2 |
| | **Gesamt:** | 11.25 | 13 |

| | |
|---|---|
| Task... | Durchzuführendes Arbeitspaket |
| Mitglied... | Bearbeitendes Teammitglied |
| Geschätze Zeit... | Geschätzte Zeit für den jeweiligen task in (Stunden) |
| Benötigte Zeit... | Tatsächlich investierte Zeit in den jeweiligen Task (Stunden) |

# 3 Datenbankentwurf

## 3.1 ERD DB1

Bei dem Entwurf der Datenbank wurde auf ein simples Beispiel, bestehend aus drei Tabellen, zurückgegriffen. Es handelt sich um ein Buch, dass von einem bestimmtem Autor verfasst wurde, von dem eine eindeutige ID, der Name und das Geburtsdatum bekannt ist.
Jedes Buch wird von einem Verlag verlegt, der ebenfalls über eine eindeutige ID und einem Namen besitzt.



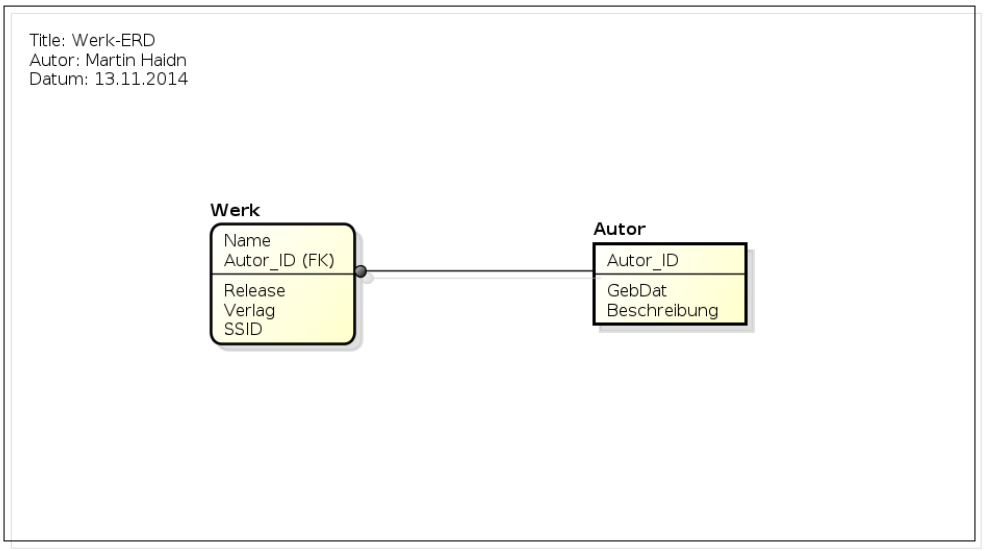

### 3.1.1 Create-Script

Das Create-Script wurde über das Export-Tool von Astah an Hand des ER-Diagramms erstellt.

```
CREATE TABLE Autor (
Autoren ID INT NOT NULL,
Name VARCHAR(25),
Geburtsdatum DATE
);

ALTER TABLE Autor ADD CONSTRAINT PK_Autor PRIMARY KEY (Autoren ID);


CREATE TABLE Verlag (
ID INT NOT NULL,
Verlagsname VARCHAR(30)
);

ALTER TABLE Verlag ADD CONSTRAINT PK_Verlag PRIMARY KEY (ID);


CREATE TABLE Buch (
Internationale Standard Buchnummer VARCHAR(20) NOT NULL,
Buchtitel VARCHAR(30),
Autoren ID INT NOT NULL,
Verlags ID INT
);

ALTER TABLE Buch ADD CONSTRAINT PK_Buch PRIMARY KEY (Internationale Standard Buchnummer);


ALTER TABLE Buch ADD CONSTRAINT FK_Buch_0 FOREIGN KEY (Autoren ID) REFERENCES Autor (Autoren ID);
ALTER TABLE Buch ADD CONSTRAINT FK_Buch_1 FOREIGN KEY (Verlags ID) REFERENCES Verlag (ID);
```

## 3.2 ERD DB2

Ähnlich wie bei der ersten Datenbank, enthält die Zweite, Tabellen über das Halten von Informationen von Büchern, allerdings besitzt der Verlag keine eigene Identität mehr.




### 3.2.1 Create-Script

Das Create-Script wurde über das Export-Tool von Astah and Hand des ER-Diagrammes erstellt.

```
CREATE TABLE Autor (
id INT NOT NULL,
GebDat DATE,
Beschreibung CHAR(10)
);

ALTER TABLE Autor ADD CONSTRAINT PK_Autor PRIMARY KEY (id);


CREATE TABLE Werk (
name VARCHAR(50) NOT NULL,
id INT NOT NULL,
release DATE,
verlag VARCHAR(50),
SSID CHAR(10)
);

ALTER TABLE Werk ADD CONSTRAINT PK_Werk PRIMARY KEY (name,id);


ALTER TABLE Werk ADD CONSTRAINT FK_Werk_0 FOREIGN KEY (id) REFERENCES Autor (id);
```

# 4 Middleware

# 5 Arbeitsdurchführung

Für das Mappen auf die jeweiligen Datenbanken soll auf propel zurückgegriffen werden, ein Tool das letztes Jahr im Zuge einer Übung zum objektrelationalen Mapping verwendet wurde.
Der Vorteil der sich daraus ergibt liegt darin, dass die PHP-Klassen für das Mappen in die Datenbanken automatisch aus einem XML-Schema generiert werden und so nur noch die Synchronisation dazwischen implementiert werden muss.

Da die Installation von Propel über den Peer-PHP-Installer um einiges zügiger geht, wird dieser vorher installiert.

## 5.1 Peer Installation

Unter Linux kann Peer einfach mit dem folgenden Befehl installiert werden:

```
apt-get install php5-xsl php-pear
```

Da für manche Anwendungsfälle Phing benötigt wird macht es Sinn, auch Dies noch zu installieren:

```
sudo pear channel-discover pear.phing.info
sudo pear install phing/phing
```

## 5.2 Propel Installation

Sind die vorherigen Punkte geglückt, kann nun Propel installiert werden:

```
pear channel-discover pear.propelorm.org
pear install -a propel/propel_generator
```

Da für das Generieren der PHP-Klassen der Generator ausreicht, ignorieren wir die Propel-Runtime.

Bei der Installation von Propel sind leider einige Probleme aufgetreten, die Zeit und Nerven kosteten. Es empfiehlt sich nicht den Peer-Installer zu verwenden, sondern Propel über den Package-Manager zu Installieren, bzw. die Sources handisch zu laden.

## 5.3 Generieren der PHP-Klassen

Propel bietet die Funktion die PHP-Klassen für das Mapping in die Datenbanken an Hand eines XML-Shemas zu generieren.
Für diesen Zweck wurden für beide Seiten ein XML-Shema erstellt. Zusätzlich wurde zu jeder Tabelle eine Synchronisationstabelle erstellt, die neben dem Datensätzen ein Enum der Änderung beinhalten.
(INSERT, UPDATE, DELETE)

### MySql

```xml
<?xml version="1.0" encoding="UTF-8"?>
<database name="Buch" defaultIdMethod="native">

  <table name="autor" phpName="Autor">
    <column name="id" phpName="ID" type="integer" required="true" primaryKey="true" autoIncrement="true"/>
    <column name="name" phpName="Name" type="varchar" size="50" required="true" />
    <column name="gebdat" phpName="GebDat" type="date" required="true" />
  </table>

  <table name="verlag" phpName="Verlag">
    <column name="id" phpName="ID" type="integer" required="true" primaryKey="true" autoIncrement="true"/>
    <column name="name" phpName="Name" type="varchar" size="50" required="true" />
  </table>

  <table name="buch" phpName="Buch">
    <column name="isbn" phpName="ISBN" type="varchar" size="24" required="true" primaryKey="true" />
    <column name="titel" phpName="Titel" type="varchar" size="50" required="true" />
    <column name="autor_id" phpName="Autor_ID" type="integer" />
    <column name="verlag_id" phpName="Verlag_ID" type="integer" />

    <foreign-key foreignTable="autor" phpName="Autor">
      <reference local="autor_id" foreign="id"/>
    </foreign-key>
    <foreign-key foreignTable="verlag" phpName="Verlag">
      <reference local="verlag_id" foreign="id"/>
    </foreign-key>
  </table>

  <!-- Tabllen fuer die Synchronisation -->
  <!-- Sync-Table autor -->
  <table name="autor_changes" phpName="Autor_Changes">
    <column name="id" phpName="ID" type="integer" required="true" primaryKey="true" autoIncrement="true"/>
    <column name="name" phpName="Name" type="varchar" size="50" required="true" />
    <column name="gebdat" phpName="GebDat" type="date" required="true" />
    <column name="time_stamp" phpName="Time_Stamp" type="date" required="true" />
    <column name="action" phpName="Action" type="varchar" size="3" sqlType="enum('INSERT', 'UPDATE', 'DELETE')" required="true" />
  </table>

  <!-- Sync-Table verlag -->
  <table name="verlag_changes" phpName="Verlag_Changes">
    <column name="id" phpName="ID" type="integer" required="true" primaryKey="true" autoIncrement="true"/>
    <column name="name" phpName="Name" type="varchar" size="50" required="true" />
    <column name="time_stamp" phpName="Time_Stamp" type="date" required="true" />
    <column name="action" phpName="Action" type="varchar" size="3" sqlType="enum('INSERT', 'UPDATE', 'DELETE')" required="true" />
  </table>

  <!-- Sync-Table buch -->
  <table name="buch_changes" phpName="Buch_Changes">
    <column name="isbn" phpName="ISBN" type="varchar" size="24" required="true" primaryKey="true" />
```

### Postgres

```xml
<?xml version="1.0" encoding="UTF-8"?>
<database name="Werk" defaultIdMethod="native">

  <table name="autor" phpName="Autor">
    <column name="id" phpName="ID" type="integer" required="true" primaryKey="true" autoIncrement="true"/>
    <column name="gebdat" phpName="GebDat" type="varchar" size="50" required="true" />
    <column name="beschreibung" phpName="Beschreibung" type="varchar" required="false" />
  </table>

  <table name="werk" phpName="Werk">
    <column name="name" phpName="Name" type="varchar" required="true" primaryKey="true" />
    <column name="autor_id" phpName="Autor_ID" type="integer" />
    <column name="isbn" phpName="ISBN" type="varchar" size="24" required="true" />
    <column name="release" phpName="ReleaseDate" type="date" required="true" />

    <foreign-key foreignTable="autor" phpName="Autor">
      <reference local="autor_id" foreign="id"/>
    </foreign-key>
  </table>

  <!-- Tabllen fuer die Synchronisation -->
  <!-- Sync-Table autor -->
  <table name="autor_changes" phpName="Autor_Changes">
    <column name="id" phpName="ID" type="integer" required="true" primaryKey="true" autoIncrement="true"/>
    <column name="gebdat" phpName="GebDat" type="varchar" size="50" required="true" />
    <column name="beschreibung" phpName="Beschreibung" type="varchar" required="false" />
    <column name="time_stamp" phpName="Time_Stamp" type="date" required="true" />
    <column name="action" phpName="Action" type="varchar" size="3" sqlType="enum('INSERT', 'UPDATE', 'DELETE')" required="true" />
  </table>

  <!-- Sync-Table werk -->
  <table name="werk_changes" phpName="Werk_Changes">
    <column name="name" phpName="Name" type="varchar" required="true" primaryKey="true" />
    <column name="autor_id" phpName="Autor_ID" type="integer" />
    <column name="isbn" phpName="ISBN" type="varchar" size="24" required="true" />
    <column name="release" phpName="ReleaseDate" type="date" required="true" />
    <column name="time_stamp" phpName="Time_Stamp" type="date" required="true" />
    <column name="action" phpName="Action" type="varchar" size="3" sqlType="enum('INSERT', 'UPDATE', 'DELETE')" required="true" />

    <foreign-key foreignTable="autor" phpName="Autor">
      <reference local="autor_id" foreign="id"/>
    </foreign-key>
  </table>

</database>
```

Mit dem Befehl "*propel gen*" können die PHP-Klassen, unter der Verwendung des XML-Shemas, das auf der gleichen Ebene liegen muss, erstellt werden. Diese können nun im Unterverzeichnis "**build**" gefunden werden.

## 5.4 Umsetzen der Middleware

Um die beiden Datenbanken synchronisieren zu können, wurden auf alle Tabellen drei Trigger gesetzt, die jeweils entweder auf einen Insert, Update, oder Delete warten.
Werden diese Trigger ausgelöst, so wird abhängig von der Änderung der neue Satz in die Synchronisationstabelle geschrieben.
Die Middleware selektiert in einem definierten, möglichst kurzem, Zeitintervall die Synchronisationstabellen der Datenbanken und bringt so beide Seiten wieder auf den gleichen Stand.
Nach der Synchronisation werden die Datensätze aus den Syc-Tabellen wieder entfernt.

### Trigger definition

```
-- Trigger-Definition fuer die Synchronisation
-- Buch (MySQL)

--autor
DELIMITER //
--DROP TRIGGER insert_autor;
CREATE TRIGGER insert_autor AFTER INSERT ON autor
    FOR EACH ROW BEGIN
        INSERT INTO autor_changes (id, name, gebdat, time_stamp, action) VALUES (NEW.id, NEW.name, NEW.gebdat, NOW(), "INSERT");
    END; //
DELIMITER ;

DELIMITER //
----DROP TRIGGER delete_autor;
CREATE TRIGGER delete_autor AFTER DELETE ON autor
    FOR EACH ROW BEGIN
        INSERT INTO autor_changes (id, name, gebdat, time_stamp, action) VALUES (OLD.id, OLD.name, OLD.gebdat, NOW(), "DELETE");
    END; //
DELIMITER ;

DELIMITER //
--DROP TRIGGER update_autor;
CREATE TRIGGER update_autor AFTER UPDATE ON autor
    FOR EACH ROW BEGIN
        INSERT INTO autor_changes (id, name, gebdat, time_stamp, action) VALUES (NEW.id, NEW.name, NEW.gebdat, NOW(), "UPDATE");
    END; //
DELIMITER ;

--verlag
DELIMITER //
--DROP TRIGGER insert_verlag;
CREATE TRIGGER insert_verlag AFTER INSERT ON verlag
    FOR EACH ROW BEGIN
        INSERT INTO verlag_changes (id, name, time_stamp, action) VALUES (NEW.id, NEW.name, NOW(), "INSERT");
    END; //
DELIMITER ;

DELIMITER //
--DROP TRIGGER delete_verlag;
CREATE TRIGGER delete_verlag AFTER DELETE ON verlag
    FOR EACH ROW BEGIN
        INSERT INTO verlag_changes (id, name, time_stamp, action) VALUES (OLD.id, OLD.name, NOW(), "DELETE");
    END; //
DELIMITER ;

DELIMITER //
--DROP TRIGGER update_verlag;
CREATE TRIGGER update_verlag AFTER UPDATE ON verlag
```