Asset Management

Datenbank Projektarbeit

Juan Bernstein & Oliver Czabala

Inhalt

[1. Aufgabenbeschreibung 3](#_Toc11324760)

[2. ER-Modell 4](#_Toc11324761)

[3. Bestimmung der Entitäten 5](#_Toc11324762)

[4. ER-Diagramm 6](#_Toc11324763)

[5. Erstellung der Entitäten 6](#_Toc11324764)

[6. Dateien einlesen / CSV Import 8](#_Toc11324765)

[Script 8](#_Toc11324766)

[7. Views 9](#_Toc11324767)

[View 1 9](#_Toc11324768)

[Script 9](#_Toc11324769)

[View 2 9](#_Toc11324770)

[Script 9](#_Toc11324771)

[View 3 10](#_Toc11324772)

[Script 10](#_Toc11324773)

[View 4 10](#_Toc11324774)

[Script 10](#_Toc11324775)

[8. Rights Management 11](#_Toc11324776)

[Script 11](#_Toc11324777)

[9. Stored Procedures 12](#_Toc11324778)

[Neuer Mitarbeiter anlegen 12](#_Toc11324779)

[Script 12](#_Toc11324780)

[Neues Asset erfassen 13](#_Toc11324781)

[Script 13](#_Toc11324782)

[10. Anhang 13](#_Toc11324783)

# Aufgabenbeschreibung

Der Kunde wünscht sich eine neue Datenbank um seine Arbeitsplätze zu inventarisieren.  
In diesem Inventarsystem soll ersichtlich sein, wo ein Mitarbeitender seinen Arbeitsplatz hat und welche Geräte ihm zugewiesen worden sind.

Ein Arbeitsplatz besteht meistens aus einem Notebook, einem Monitor sowie Peripherie.

Die Modelle der Geräte sind genormt, daher gibt es viele Geräte die vom selben Modell sind, jedoch ein abweichendes Anschaffungsdatum somit auch ein abweichendes Garantieablaufdatum.

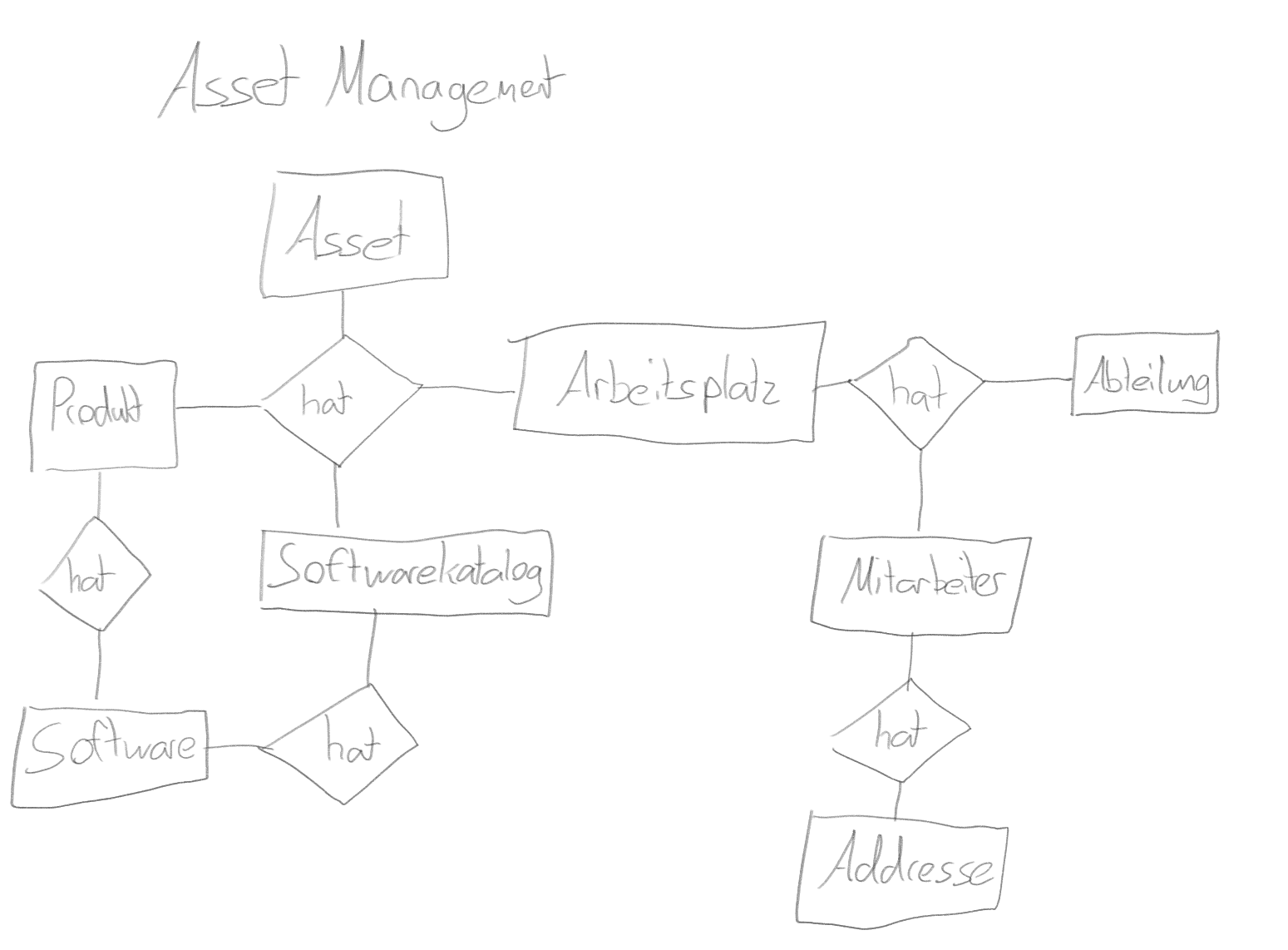
Um kosten bei der Softwarebeschaffung einzusparen, soll ebenfalls möglich sein einem Mitarbeiter verschiedene Software zuzuordnen.

Im Firmengebäude gibt es verschiedene Abteilungen welche zugewiesene Softwaregruppen besitzen.

Dem Kunden soll ermöglicht werden, Daten schnell auslesen zu können, Beispiele hierzu sind: z.B. zu sehen wann welche Geräte ersetzt würden müssen bei einem LifeCycle von 3 Jahren, sowie auch eine Statistik wie viele Mitarbeiter welche Software benutzen.

Es soll auch ersichtlich sein wann welche Software in welcher Softwareversion installiert worden ist, um zu sehen welche Geräte aktualisiert werden müssen.

# ER-Modell



Beschreibung:

Das Asset Management ist folgendermassen unterteilt:  
Die Hauptkategorien die wir bewirtschaften sind Mitarbeiter, sowie dazugehörige Assets.

Die Assets selber werden einem Arbeitsplatz zugeordnet, solch ein Arbeitsplatz hat zum Beispiel einen Laptop, einen Monitor, Eingabegeräte sowie eine Dockingstation.

Dem Asset selber ist ebenfalls Software zugeordnet, damit wir eine Kontrolle haben welche Software auf welchem Gerät installiert ist.

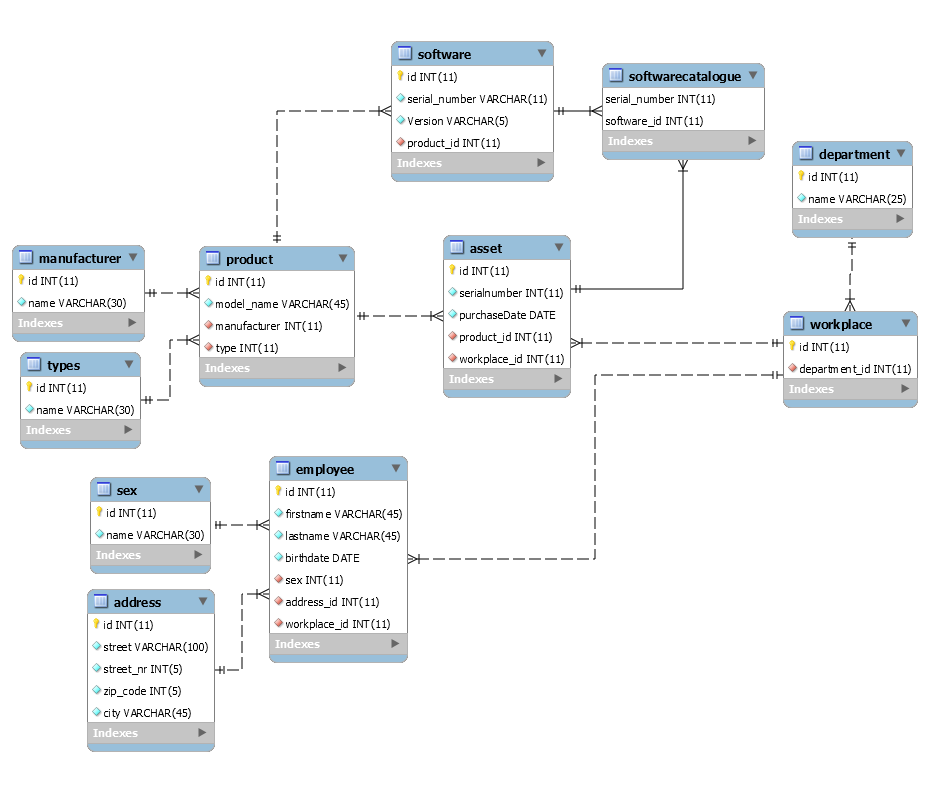
Mitarbeiter haben einen festen Arbeitsplatz / Büro und solch ein Arbeitsplatz ist einer Abteilung zugeordnet. Ein Mitarbeiter hat jedoch auch eine Adresse sowie persönliche Informationen.

# Bestimmung der Entitäten

Die folgenden Entitäten wurden mit der Aufgabenbeschreibung erarbeitet.

|  |  |  |
| --- | --- | --- |
| Entität | Beschreibung | Beispiel |
| Product | Produkte werden hier erfasst | HP Elitebook 830 G5 |
| Software | Verfügbare Software | Microsoft Office 365 |
| Asset | Assets sind die Eigenschaften einzelner Geräte | Kaufdatum, Garantieablauf |
| Software Catalogue | Softwaregruppen welche zugeordnet werden können | Softwaregruppe Human Ressources |
| Workplace | Arbeitsplatz zugeordnet an User mit diversen Assets | Büro 1.01 |
| Employee | Daten zum Mitarbeiter | Hans Mustermann |
| Department | Institute / Abteilung | Human Ressources |
| Address | Addresse der Employees | Hans Muster, Musterstrasse 1, 1111 Musterort |

# ER-Diagramm



Beschreibung:

Das ER-Diagramm wurde auf Basis von der erstellten Datenbank erstellt, dies wurde mit dem Programm MySQL Workbench per Reverse Engineering erstellt.

1. Erstellung der Entitäten

DROP DATABASE IF EXISTS Assetmgmt;

CREATE DATABASE Assetmgmt;

USE Assetmgmt;

CREATE TABLE product (

id INT NOT NULL AUTO\_INCREMENT,

model\_name VARCHAR(45) NOT NULL,

manufacturer INT NOT NULL,

type INT NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE manufacturer (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(30) NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE types (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(30) NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE asset (

id INT NOT NULL AUTO\_INCREMENT,

serialnumber INT NOT NULL,

purchaseDate DATE NOT NULL,

product\_id INT NOT NULL,

workplace\_id INT NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE software (

id INT NOT NULL AUTO\_INCREMENT,

serial\_number varchar(11) not null,

Version VARCHAR(5) NOT NULL,

product\_id INT NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE softwarecatalogue (

serial\_number INT NOT NULL,

software\_id INT NOT NULL

);

CREATE TABLE workplace (

id INT NOT NULL AUTO\_INCREMENT,

department\_id INT NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE department (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(25) NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE employee (

id INT NOT NULL AUTO\_INCREMENT,

firstname VARCHAR(45) NOT NULL,

lastname VARCHAR(45) NOT NULL,

birthdate DATE NOT NULL,

sex INT NOT NULL,

address\_id INT NOT NULL,

workplace\_id INT NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE sex (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(30) NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE address (

id INT NOT NULL AUTO\_INCREMENT,

street VARCHAR(100) NOT NULL,

street\_nr INT(5) NOT NULL,

zip\_code INT(5) NOT NULL,

city VARCHAR(45) NOT NULL,

PRIMARY KEY (id)

);

ALTER TABLE product ADD FOREIGN KEY (manufacturer) REFERENCES manufacturer(id);

ALTER TABLE product ADD FOREIGN KEY (type) REFERENCES types(id);

ALTER TABLE asset ADD FOREIGN KEY (product\_id) REFERENCES product(id);

ALTER TABLE asset ADD FOREIGN KEY (workplace\_id) REFERENCES workplace(id);

ALTER TABLE software ADD FOREIGN KEY (product\_id) REFERENCES product(id);

ALTER TABLE softwarecatalogue ADD constraint primary key (serial\_number,software\_id);

ALTER TABLE softwarecatalogue ADD FOREIGN KEY (serial\_number) REFERENCES asset(id);

ALTER TABLE softwarecatalogue ADD FOREIGN KEY (software\_id) REFERENCES software(id);

ALTER TABLE employee ADD FOREIGN KEY (workplace\_id) REFERENCES workplace(id);

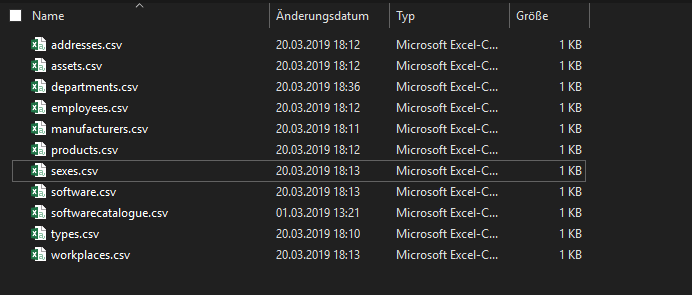
ALTER TABLE employee ADD FOREIGN KEY (sex) REFERENCES sex(id);

ALTER TABLE employee ADD FOREIGN KEY (address\_id) REFERENCES address(id);

ALTER TABLE workplace ADD FOREIGN KEY (department\_id) REFERENCES department(id);

# Dateien einlesen / CSV Import

Im Script werden folgende Dateien eingelesen:



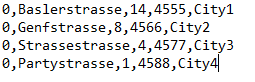
Die CSV Dateien sollten im Folder:

«C:/CSV» gespeichert werden, dort werden sie im Script importiert.

Die CSV Dateien werden kommagetrennt eingelesen, sollten Fehler bei der Eingabe entstehen entsteht eine Fehlermeldung welche verständlicherweise den Fehler ausgibt, Beispiel hierfür ist eine Meldung wie: «Wert fehlt», das kann dadurch passieren, dass ein Wert zu wenig hineingeschrieben worden ist.

Die Trennung geschieht via Kommazeichen und ein neuer Satz wird mit einer neuen Zeile (Break) voneinander getrennt.

Beispiel Inhalt anhand der Datei addresses.csv:



## Script

/\*Insert Data\*/

LOAD DATA INFILE 'C:/CSV/types.csv'

INTO TABLE types

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/CSV/manufacturers.csv'

INTO TABLE manufacturer

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/CSV/products.csv'

INTO TABLE product

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/CSV/software.csv'

INTO TABLE software

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/CSV/addresses.csv'

INTO TABLE address

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/CSV/departments.csv'

INTO TABLE department

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/CSV/workplaces.csv'

INTO TABLE workplace

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/CSV/sexes.csv'

INTO TABLE sex

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/CSV/employees.csv'

INTO TABLE employee

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/CSV/assets.csv'

INTO TABLE asset

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/CSV/softwarecatalogue.csv'

INTO TABLE softwarecatalogue

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

# Views

## View 1

Die erste View zeigt auf einem Blick eine Hardwareliste welche Hersteller sowie Modellname beinhaltet.

### Script

create view Hardware\_List as

select manufacturer.name as 'Hersteller', product.model\_name as 'Modellname', asset.serialnumber as 'Serialnumber' from product

join manufacturer on manufacturer.id

join asset on asset.id

where product.id = asset.product\_id and product.manufacturer = manufacturer.id;

## View 2

Die zweite View erzeugt eine Auflistung aller Hardware die ein bestimmter Mitarbeiter zugeordnet hat.

### Script

create view Employee\_Hardware\_List as

select employee.firstname as 'Firstname', employee.lastname as 'Lastname', asset.serialnumber as 'Seriennummer', product.model\_name as 'Modell' from asset

join product on product.id

join workplace on workplace.id

join employee on employee.id

where asset.product\_id = product.id and asset.workplace\_id = workplace.id and employee.workplace\_id = workplace.id;

## View 3

Die dritte View erzeugt eine Auflistung welche Mitarbeiter in einem Departement welche Assets hat.

### Script

create view Departement\_Employee\_List as

select department.name as 'Department', employee.firstname as 'Firstname', employee.lastname as 'Lastname', workplace.id as 'Büro', asset.serialnumber as 'Asset', product.model\_name as 'Modell' from employee

join workplace on workplace.id

join department on department.id

join asset on asset.id

join product on product.id

where employee.workplace\_id = workplace.id and workplace.department\_id = department.id and asset.workplace\_id = workplace.id and asset.product\_id = product.id;

## View 4

Die vierte View zeigt an welche Software am meisten von einem bestimmten Departement genutzt wird.

### Script

create view Most\_Used\_Software as

select department.name as 'Department', product.model\_name as 'Software Name', count(product.id) as 'Anzahl' from department

join workplace on workplace.id

join asset on asset.id

join softwarecatalogue on softwarecatalogue.serial\_number

join software on software.id

join product on product.id

where department.id = workplace.department\_id

and asset.workplace\_id = workplace.id

and asset.id = softwarecatalogue.serial\_number

and softwarecatalogue.software\_id = software.id

and software.product\_id = product.id

group by department.name;

# Rights Management

Damit nicht alle Benutzer immer volle Rechte besitzen, haben wir uns für einen Approach entschieden bei dem es verschiedene Benutzergruppen gibt.



* DB Administrator: Hat volle Berechtigungen
* Human Ressources, Softwaremanager sowie Hardware Manager: Sind dem DB Administrator direkt unterstellt und haben auf ihre benötigten Orte Berechtigungen.
* Guest: Kann sehen wo ein Mitarbeiter seinen Arbeitsplatz hat.

Beispiel im Script für einen User in den Human Ressources Abteilung:

## Script

DROP USER IF EXISTS hr@’localhost’;

create user hr@’localhost’ identified by '123';

grant all privileges on employee to hr@’localhost’;

grant all privileges on address to hr@’localhost’;

grant all privileges on sex to hr@’localhost’;

grant all privileges on department to hr@’localhost’;

grant all privileges on workplace to hr@’localhost’;

# Stored Procedures

## Neuer Mitarbeiter anlegen

Mit dieser Prozedur soll ein neuer Mitarbeiter erfasst werden, benötigte Attribute sind folgende:

Entität **Employee**

|  |  |  |
| --- | --- | --- |
| **Attribut** | **Datentyp** | **Beschreibung** |
| Firstname | VARCHAR(45) | Vorname |
| Lastname | VARCHAR(45) | Nachname |
| Sex | ENUM | Geschlecht |
| Birthday | DATE | Geburtsdatum |

Entität **Address**

|  |  |  |
| --- | --- | --- |
| **Attribut** | **Datentyp** | **Beschreibung** |
| Street | VARCHAR(100) | Adresse |
| HouseNr | VARCHAR(5) | Hausnummer |
| PLZ | INT | Postleitzahl |
| City | VARCHAR(45) | Stadt / Dorf |

### Script

DROP procedure IF EXISTS `newEmployee`;

DELIMITER //

CREATE PROCEDURE newEmployee(

Firstname varchar (45),

Lastname varchar (45),

Sex varchar(15),

Birthday date,

Street varchar(100),

HouseNr int,

PLZ int,

City varchar(45))

BEGIN

insert into address (street,street\_nr,zip\_code,city)

values (Street,HouseNr,PLZ,City);

select @adressID := a.id from address as a

where a.street = Street AND a.zip\_code = PLZ

Limit 1;

select @sexID := s.id from sex as s

where s.name = Sex;

select @workplace\_id := w.id from workplace as w

join employee as e on w.id <> e.workplace\_id

group by e.id limit 1;

insert into employee (firstname,lastname,birthdate,sex,address\_id,workplace\_id)

values (Firstname,Lastname,Birthday,@sexID,@adressID,@workplace\_id);

END//

DELIMITER ;

## Neues Asset erfassen

Mit dieser Prozedur wird ein neues Asset angelegt mit dem Kaufdatum

Entität **Asset**

|  |  |  |
| --- | --- | --- |
| **Attribut** | **Datentyp** | **Beschreibung** |
| Serialnumber | INT | Seriennummer des Gerätes |
| PurchaseDate | DATE | Kaufdatum |

### Script

DROP procedure IF EXISTS `newAsset`;

DELIMITER //

CREATE PROCEDURE newAsset(

Firstname varchar (45),

Lastname varchar (45),

Serialnumber int(11),

BuyDate date,

ModellName varchar(45))

BEGIN

select @productID := p.id from product as p

where p.model\_name = ModellName;

select @workplaceID := e.workplace\_id from employee as e

where e.firstname = Firstname AND e.lastname = Lastname;

insert into asset (serialnumber,purchaseDate,product\_id,workplace\_id)

values (Serialnumber,BuyDate,@productID,@workplaceID);

END//

DELIMITER ;

# Anhang

-MySQL Script als Datei «Database\_Asset.sql»

-ER Diagramm als Bilddatei