Hochschule Mannheim/
Fakultät für Informatik

Version 0.3.x

21.09.2023

DBA CompanyDemo

Ein vollständiges Beispiel
einer Datenbankanwendung
für Unternehmensdaten
für relationale,
objekt-orientierte und
dokument-orientierte Datenbanken

Inhalt

1 Über das Dokument

Dieses Dokument beschreibt, wie DBACompanyDemo in der Version 0.3.x verwendet werden kann. Der zugehörige Moodle-Kurs ist https://moodle.hs-mannheim.de/course/view.php?id=548

Wichtig: Die Dokumentation ist an etlichen Stellen noch nicht vollständig und möglicherweise auch inkonsistent. Hier werden gerne Freiwillige gesucht, die dies im zugehörigen Github-Projekt verbessern. Wie das geht, ist weiter unten beschrieben.

Die wichtigsten bekannten Fehler (siehe auch https://github.com/informatik-mannheim/dbacompanydemo/issues) sind:

- MongoDB liefert Null-Pointer-exceptions

2 Einführung

2.1 Zweck

Vielen Dank, dass Sie das DBA-Firmenbeispiel heruntergeladen haben. Dieses Dokument beschreibt die Beispiel-Firmendatenbank-Anwendung. Sie soll Studierenden die Möglichkeit bieten, die in der Vorlesung DBA gezeigten Methoden an einem Beispiel zu vertiefen. Die Anwendung umfasst eine komplette relationale Datenbank (MySQL), eine objektorientierte Datenbank (DB4O) sowie die Nutzung dieser beiden Datenbanken aus einer Programmiersprache heraus (Java und später Scala). Sie sehen

- wie mit Java Database Connectivity (JDBC) und SQL-Statements aus Java heraus eine relationale DB genutzt wird,
- wie SQL-Abfragen auf die Beispieldaten gemacht werden,
- wie sich mit Hilfe des Objekt-Relationalen-Mappers Hibernate relationale Datenbanken noch einfacher nutzen lassen,
- wie eine objektorientierte DB aus Java heraus genutzt wird und wie sich der Aufwand für die Programmierung deutlich verringert,
- wie sich mit einer einfachen Konsolen-Anwendung die Daten bearbeiten lassen. Diese Anwendung ist 100% unabhängig von der den o.g. genannten Technologien (d.h. JDBC, MySQL, Hibernate, DB4O usw.)

Nachdem Sie das Programmpaket heruntergeladen haben, haben Sie folgende Möglichkeiten:

1. Sie nutzen einfach nur die Konsolen-Anwendung, die auf eine zentrale DB der Hochschule zugreift. Ein Doppelklick auf die Anwendung (`companydemo.bat`) genügt hierfür.
2. Sie nutzen die Konsolen-Anwendung, tauschen jedoch die relationale DB-Instanz mit einer lokalen Installation aus. Damit können Sie die Daten beliebig verändern und sind Herr Ihrer Daten.
3. Sie nutzen die Konsolen-Anwendung und speichern Ihre Daten lokal in der objekt-orientierten DB DB4O.
4. Sie erweitern die Konsolen-Anwendung oder das Datenbank-Schema, indem sie den mitgelieferten Source-Code verändern. Sie entscheiden, ob die Kon-

solen-Anwendung „manuell“ über JDBC oder mittels Hibernate auf die relationale DB MySQL zugreift uvm.

Zukünftig sehen Sie noch

- wie Scala und die objektorientierte DB den Aufwand für Softwareentwicklung abermals reduzieren.

Der Source-Code ist open-source, d.h. Sie als Studierende sind herzlich eingeladen, das Beispiel weiter zu entwickeln und etwaige Fehler zu beheben. Ihre Änderungen werden bestimmt übernommen. Wie das geht, erfahren Sie im Abschnitt 3.3.1.

2.2 Warum sich bestimmte Technologien lohnen

Schauen wir uns mal an, wie viele Lines of Code (Anzahl Zeilen im Source-Code, wobei SQL-Skripte nicht gezählt wurden) nötig sind, um die Konsolen-Anwendung zu realisieren. Hinweis: Die verwendete Metrik zählt nur Anweisungen (z.B. `i = 1;`) aber keine Deklarationen (z.B. `int i;`) oder Kommentare.

				Lösung		
Bereich	**Anzahl Klassen**	**LOC**		**JDBC**	**Hibernate**	**DB4O**
User-Interface	3	699		699	699	699
Fachliche Modell	10	1.231		1.231	1.231	1.231
Generische DB-Anbindung	4	294		294	294	294
JDBC-Implementierung	12	1.467		1.467		
Hibernate-Implementierung	3	455			455	
DB4O-Implementierung	3	498				498
Gesamt	35	4.644		3.691	2.679	2.722
				100%	73%	74%

Ganz offensichtlich macht der Hibernate- bzw. DB4O-Anteil nur ein Drittel des JDBC-Umfangs aus – wer hier weniger zu codieren hat, macht weniger Fehler und ist sowieso schneller fertig. Aber warum ist das so? Bleiben Sie am Ball!

2.3 Was fehlt noch?

Obwohl die die aktuelle Version bereits jede Menge kann, fehlen leider immer noch einige Dinge, die bei einer professionellen Datenbank-Entwicklung nicht fehlen dürfen. Neben den vermutlichen immer noch vorhandenen Bugs sind dies v.a.

- Usermanagement und Zugriffsrechte

- Transaktionsmanagement
- Ressourcenmanagement der DB-Verbindungen
- Ausreichende Unit-Tests
- Ausreichende Kommentare im Source-Code. Bisher lag der Schwerpunkt auf vielen Features.

Andere Technologien wie z.B. ein Konfigurations-Framework ala Spring werden bewusst nicht eingesetzt. Näheres hierzu finden Sie in Kapitel 5.1.

3 Anwendung

Dieser Abschnitt beschreibt das Einsatzgebiet der Datenbankanwendung.

3.1 Anwendungsfälle

Dieses Fallbeispiel beschreibt Entitäten aus einem Unternehmen. Um das Beispiel möglichst einfach zu halten, soll es möglichst wenige Klassen geben und alle Klassen sind so einfach gehalten, wie gerade noch möglich, aber so umfangreich wie nötig. D.h. es fehlen möglicherweise sinnvolle Attribute usw. Dennoch sollen alle relevanten Beispiele aus der Vorlesung DBA bzw. DM damit erläutert werden können.

In einer produzierenden Firma gibt es allgemein Mitarbeiter, die sich in Arbeiter und Angestellte unterteilen. Jeder Mitarbeiter kann einer Abteilung zugeordnet sein und hat genau einen Vorgesetzten (außer der Vorstand). Ein Firmenwagen wird exklusiv von einem Angestellten benutzt, jedoch niemals exklusiv von einem Mitarbeiter oder Arbeiter. Nicht jeder Angestellte hat einen Wagen, meistens sind es Führungskräfte. Es gibt auch Firmenwagen ohne Zuordnung eines Angestellten, die als so genannte Pool-Fahrzeuge stundenweise gebucht und gefahren werden können. Ausschließlich Angestellte können an Projekten mitwirken. Projekte werden durch Statusberichte dokumentiert.

Typische Anwendungsfälle für dieses Beispiel können sein:

- Eine Mitarbeiterin der Personalabteilung möchte einen neuen Mitarbeiter in die Firmendatenbank eintragen.
- Ein Mitarbeiter wird einem neuen Projekt zugeteilt. Die Verwaltung möchte dies in der Firmendatenbank eintragen.
- Ein Mitarbeiter meldet der Personalabteilung eine neue Wohnadresse. Eine Mitarbeiterin der Personalabteilung möchte die Information über die Adresse in der Firmendatenbank für diesen Mitarbeiter aktualisieren.
- Ein Mitarbeiter wird entlassen. Die Personalabteilung möchte seine Daten aus der Firmendatenbank löschen.
- Eine Mitarbeiterin der Personalabteilung möchte die Datenbank nach bestimmten Mitarbeitern durchsuchen. Dazu möchte sie entweder die Personalnummer oder Name und Vorname des Mitarbeiters als Suchparameter übergeben werden.

- Die Firma ruft ein neues Projekt ins Leben. Der Projektleiter möchte deshalb ein neues Projekt in der Datenbank anlegen.
- Ein Projektleiter muss stets die Fortschritte im Projekt dokumentieren. Dazu muss er Statusberichte für sein Projekt anlegen und diese ggf. auch wieder verändern können.
- Die Firma kauft 5 neue Firmenwagen. Diese sollen in die Datenbank der Firma eingetragen werden.
- Ein Angestellter hat einen Firmenwagen beantragt und genehmigt bekommen. Die Mitarbeiterin, welche den Bestand an Firmenwagen verwaltet, will einen bestimmten Wagen diesem Mitarbeiter zuordnen.

3.2 Anforderungen an eine Beispielanwendung

Im ersten Schritt kommt das Firmenbeispiel mit einer einfachen Konsolen-Anwendung aus. Zukünftig sind auch Desktop- oder Web-Anwendungen denkbar, jedoch würden Sie den Rahmen der DB-Vorlesung sprengen.

Die Anwendung stellt dem Benutzer folgende Funktionalitäten zur Verfügung: Alle Entitäten, die weiter unten im UML- bzw. ER-Diagramm beschrieben sind, sollen neu angelegt, gesucht, verändert und ggf. gelöscht werden können. In einem ersten Schritt erfolgt keine Benutzeranmeldung – d.h. die Anwendung wird anonym betrieben.

Beispiel für einen Dialog:

```
*** Willkommen zu CompanyDemo 0.3.1 ***

Welche Datenbankzugriffsart möchtest Du nutzen?

1 SQL-Datenbank der Hochschule Mannheim (über JDBC)
2 SQL-Datenbank der Hochschule Mannheim (über Hibernate)
3 eigene SQL-Datenbank (über JDBC)
4 eigene SQL-Datenbank (über Hibernate)
5 eigene DB4O-Datenbank (Servermode; vorher startDb4oServer.* starten)
6 lokale DB4O-Datenbank
7 lokale MongoDB-Datenbank

9 Credits

0 Programm beenden
Ihre Auswahl (Zahl): 3

*** DBA-Firmenbeispiel-Hauptmenu ***

Was möchten Sie tun?
```

```
1 Personal verwalten
2 Projekte verwalten
3 Firmenwagen verwalten
4 Abteilungen verwalten

0 Zurück
Ihre Auswahl (Zahl): 1

*** Personal verwalten ***

Was möchten Sie tun?

1 Mitarbeiter/Arbeiter/Angestellter suchen (über Personalnr.)
2 Mitarbeiter/Arbeiter/Angestellter suchen (über Name, Vorname)
3 Mitarbeiter neu anlegen
4 Mitarbeiter editieren
5 Arbeiter neu anlegen
6 Arbeiter editieren
7 Angestellte neu anlegen
8 Angestellte editieren
9 Mitarbeiter/Arbeiter/Angestellter löschen
0 Zurück
Ihre Auswahl (Zahl): 2

*** Mitarbeiter suchen (über Nachname, Vorname) ***

Nachname (* am Ende möglich): L*
Vorname (* am Ende möglich): *
Fransiska Lohe (1 Angestellter)

Taste für weiter...

*** Personal verwalten ***

Was möchten Sie tun?

1 Mitarbeiter/Arbeiter/Angestellter suchen (über Personalnr.)
2 Mitarbeiter/Arbeiter/Angestellter suchen (über Name, Vorname)
3 Mitarbeiter neu anlegen
4 Mitarbeiter editieren
5 Arbeiter neu anlegen
6 Arbeiter editieren
7 Angestellte neu anlegen
8 Angestellte editieren
9 Mitarbeiter/Arbeiter/Angestellter löschen
0 Zurück
Ihre Auswahl (Zahl): 0
*** DBA-Firmenbeispiel-Hauptmenu ***

Was möchten Sie tun?

1 Personal verwalten
2 Projekte verwalten
3 Firmenwagen verwalten
4 Abteilungen verwalten
```

```
0 Zurück
Ihre Auswahl (Zahl): 2

*** Projekte verwalten ***

Was möchten Sie tun?

1 Projekt suchen (nach Bezeichnung)
2 Projekt neu anlegen
3 Projekt löschen

4 Statusberichte für Projekt ausgeben
5 Statusbericht eingeben
6 Statusbericht ändern

0 Zurück
Ihre Auswahl (Zahl): 1

*** Projekt suchen (über Bezeichnung) ***

Bezeichnung (* am Ende möglich): *

Neues Produkt entwickeln. (FOP)
---
Leute einstellen. (LES)
 Fransiska Lohe (1 Angestellter)
---

Taste für weiter...

*** Projekte verwalten ***

Was möchten Sie tun?

1 Projekt suchen (nach Bezeichnung)
2 Projekt neu anlegen
3 Projekt löschen

4 Statusberichte für Projekt ausgeben
5 Statusbericht eingeben
6 Statusbericht ändern

0 Zurück
Ihre Auswahl (Zahl): 4

*** Statusberichte für Projekt ausgeben ***

Projektkürzel: LES
Nummer:  LES.4
Datum:   17.11.11
Inhalt:  Das ist ein Statusbericht
Projekt: Leute einstellen. (LES)
Nummer:  LES.6
Datum:   18.11.11
```

```
Inhalt:  Das ist noch ein Statusbericht
Projekt: Leute einstellen. (LES)
---

Taste für weiter...

*** Projekte verwalten ***

Was möchten Sie tun?

1 Projekt suchen (nach Bezeichnung)
2 Projekt neu anlegen
3 Projekt löschen

4 Statusberichte für Projekt ausgeben
5 Statusbericht eingeben
6 Statusbericht ändern

0 Zurück
Ihre Auswahl (Zahl): 0

*** DBA-Firmenbeispiel-Hauptmenu ***

Was möchten Sie tun?

1 Personal verwalten
2 Projekte verwalten
3 Firmenwagen verwalten
4 Abteilungen verwalten

0 Zurück
Ihre Auswahl (Zahl): 0

*** Willkommen zu CompanyDemo 0.1.0 ***

Welche Datenbankzugriffsart möchten sie nutzen?

1 SQL-Datenbank der Hochschule Mannheim (über JDBC)
2 SQL-Datenbank der Hochschule Mannheim (über Hibernate)
3 eigene SQL-Datenbank (über JDBC)
4 eigene SQL-Datenbank (über Hibernate)
5 eigene DB4O-Datenbank (Servermode)
6 lokale DB4O-Datenbank

9 Credits

0 Programm beenden
Ihre Auswahl (Zahl): 0
Programm wird beendet.

Vielen Dank, dass Sie CompanyDemo genutzt haben.
Wir hoffen, dass es Spass gemacht hat.

CompanyDemo version 0.1.0, Copyright (C) 2011
CompanyDemo comes with ABSOLUTELY NO WARRANTY;
```

```
This is free software, and you are welcome
to redistribute it under certain conditions;
See gpl2.0.txt for details.

Folgende Personen haben an diesem Projekt mitgearbeitet:
 - Patrick Sturm
 - Sven Haag
 - Marius Czardybon
 - Eyuep Ünlü
 - Mike Möck
 - Markus Gumbel

Taste für weiter...
```

3.3 Installation und Start

Es gibt zwei Möglichkeiten, an das Beispiel einschließlich Source-Code zu gelangen:

- Zip-Datei. Diese beinhaltet ausführbare Programme sowie den Source-Code.
- Aktuelles Projekt über das Github-Repository https://github.com/informatik-mannheim/dbacompanydemo.git

Bitte beachten: Wenn Sie die Anwendung starten wollen, benötigen Sie vorher eine eingerichtete Datenbank. Wie das geht, ist im Abschnitt 4.5 beschrieben.

3.3.1 Lauffähiges Projekt in einer Zip-Datei

Im Moodle-Kurs der Vorlesung können Sie die aktuelle Version als Zip-Datei zum Herunterladen. Entpacken Sie die Zip-Datei `companydemo-v??.zip` (?? entspricht Versionsnummer) in ein beliebiges Unterverzeichnis. Im Wurzelverzeichnis gibt es folgende Batch-Dateien (Windows) oder Shell-Skripte (Linux):

1. `companydemo.bat / companydemo.sh`
   Dieses Programm startet die Anwendung und nutzt die bereits kompilierten Jars im Verzeichnis `./lib`.

2. `startDb4oServer.bat / startDb4oServer.sh`
   Dieser Server muss zuvor gestartet werden, wenn Sie DB4O im Servermodus betreiben wollen. Das macht Sinn, wenn Sie z.B. parallel zu Ihrer Anwendung mittels des OME auf die DB4O-Datenbank zugreifen wollen. Näheres hierzu unter ??.

3.3.2 Github-Repository und eigene Entwicklung

Das Source-Code-Konfigurationsmanagement-Tool Git erlaubt Ihnen, direkt auf den aktuellen Source-Code zuzugreifen. Hierzu müssen Sie verstehen, wie Git arbeitet. Zu Beginn ist es sicherlich einfacher, die unter Punkt 1 genannte Zip-Datei zu nutzen. Die Adresse für Github lautet:

https://github.com/informatik-mannheim/dbacompanydemo.git.

Klonen Sie einfach das Projekt auf Ihren Rechner und verändern Sie den Source-Code nach Ihren Wünschen. Pull-Request mit neuen Features werden sehr gerne entgegen genommen.

3.3.3 Projekt-Struktur

Die Verzeichnisse beider Pakete (Zip und Git) enthalten Folgendes:

- `./sql`: Alle SQL-Skripte
- `./src/java/main`: Java-Source-Code für die Anwendung selbst
- `./src/java/test`: Testfälle
- `./lib`: externe Bibliotheken (und der kompilierte Source-Code; nur in der Zip-Datei)
- `./doc`: Alle Dateien zur Dokumentation, z.B. das Handbuch und Diagramme.

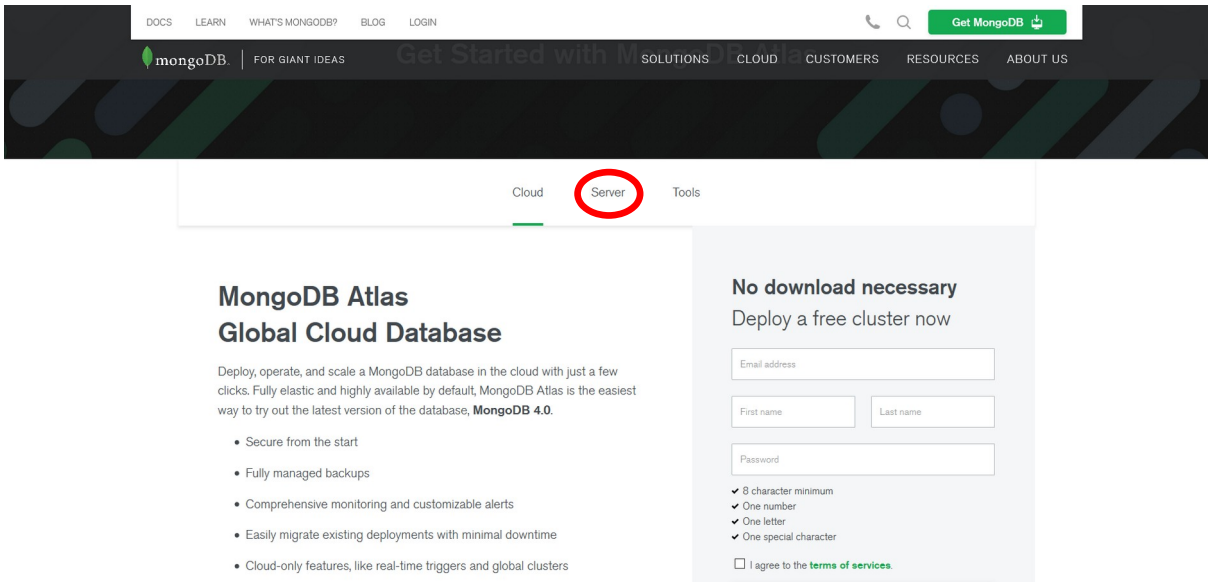
In den erstellten Unterverzeichnis (ab hier Wurzelverzeichnis `./` genannt) befinden sich eine Konfigurationsdatei für das Java-Build-Tool *ant*. Dieses Format lässt sich üblicherweise von allen bekannten Entwicklungsumgebungen (IDE) wie Visual Studio Code, IntelliJ, Eclipse usw. nutzen, um daraus ein Projekt in der IDE zu erstellen. Öffnen Sie das für Sie in Frage kommende Projekt, so dass der Java-Klassenpfad sowie die benötigten Bibliotheken (`./lib/*.jar`) verfügbar sind. Die Konsolenanwendung ist in der Klasse `net.gumbix.dba.companydemo.application.console.UI` enthalten – führen Sie die `main`-Methode aus.

Beim Start der Konsolenanwendung kann gewählt werden, welche Datenbank genutzt werden soll. Es gibt eine MariaDB, die an der Hochschule Mannheim gehostet wird[1], sowie die Möglichkeit, sich an eine beliebige eigene – beispielsweise lokale - MariaDB-Instanz zu verbinden. Die DB4O-Anbindung ist bisher nur lokal möglich, jedoch als eingebettete DB oder in der Serverversion.

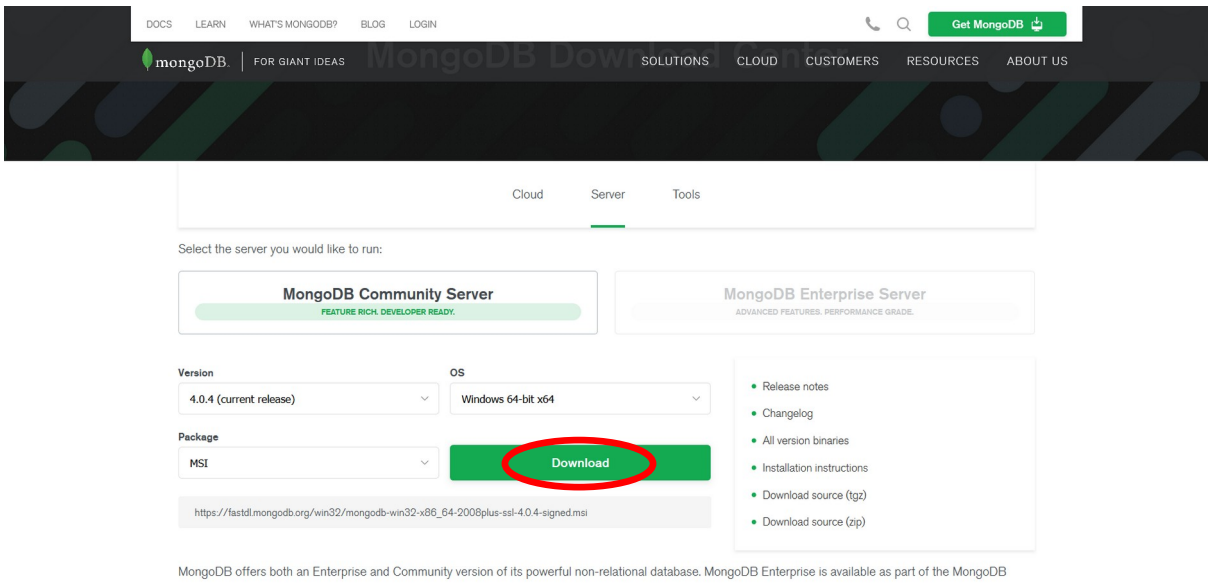
1 Der Server kann möglicherweise auch offline sein.

3.3.4 Mongo-DB installieren

Um Mongo-DB (Version 4.0.4) auf ihrem Rechner zu installieren, besuchen Sie die Website von MongoDB und installieren Sie die dort verfügbare Installationsdatei auf Ihrem Rechner (https://www.mongodb.com/download-center/community). Sie sollten in Ihrem Web-Browser nun folgendes Bild sehen.



Wählen Sie nun den Reiter „Server“ aus, der Ihnen nun folgendes Bild anzeigen sollte.



Zum Abschluss wählen Sie bitte die richtige Version (4.0.4) aus und klicken auf „Download“, um die Installations-Datei herunter zu laden. Sie müssen die Installation vollständig durchführen.

Nun können sie mit der CompanyDemo eine lokale MongoDB-Datenbank erstellen und verwalten.

4 Entwurf der Datenbank

Typisch für den Entwurf von Datenbankanwendungen ist, dass die Klassen einer objekt-orientierte Sprache mit den Tabellen einer relationalen Datenbank verbunden werden müssen. In unserem Beispiel gehen wir davon aus, dass

1. ein Klassenmodell (genauer Klassenmodell des objekt-orientierten Design, kurz OOD) vorliegt und
2. ein Datenbank-Schema,

die beide aus den Anforderungen abgeleitet wurden. Beginnen wir aber zunächst einem Modell, das lediglich die Anforderungen widerspiegelt und noch neutral ist bezüglich der späteren Anwendung in der relationalen DB oder dem Java-Programm. Dieses wird als Modell der objekt-orientierte Analyse (OOA) bezeichnet.

4.1 ER-Modell

Das folgende Entity-Relationship-Diagramm (siehe Abbildung 1) beschreibt unsere Firmenwelt. Die Entitäten, deren Attribute und Beziehungen sind noch nicht spezifisch für die spätere Implementierung.

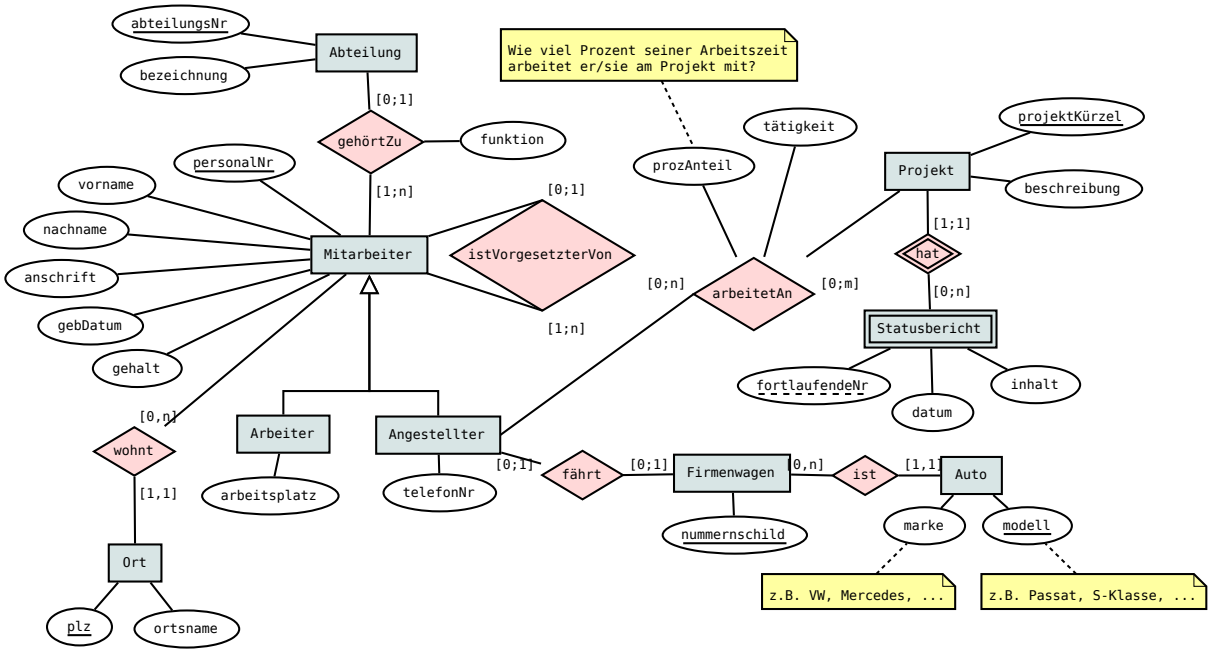


Abbildung 1: ER-Diagramm

Analog lässt sich dieses ER-Diagramm als UML-Klassendiagramm darstellen.

4.2 UML-Klassendiagram

Das UML-Diagramm (siehe Abbildung 2) ist eine 1:1-Übersetzung des ER-Diagramms.

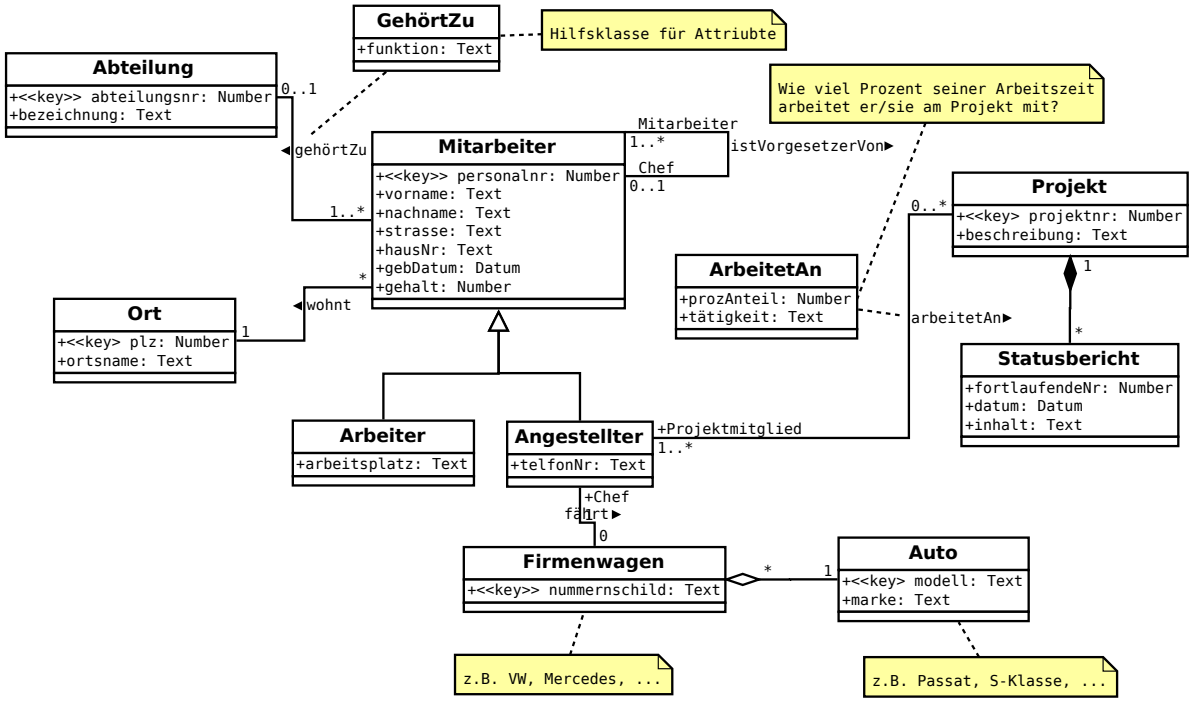


Abbildung 2: UML-Diagramm

4.3 Überführung des ER-/UML-Diagramms nach Relationen (Tabellen)

4.3.1 Entitätensrelationen (zunächst)

a) Firmenwagen (<u>nummerschild</u>, marke, modell)

b) Mitarbeiter (personalNr, vorname, nachname, anschrift, gebDatum, gehalt)

c) Arbeiter (personalNr, arbeitsplatz)

d) Angestellter (personalNr, telefonNr)

e) Abteilung (abteilungsNr, bezeichnung)

f) Projekt (projektId, bezeichnung)

g) StatusBericht (projektId#, fortlaufendeNr, datum, inhalt)
Da StatusBericht eine schwache Entität ist, muss die projektId der starken Entität mit in Primärschlüssel aufgenommen werden.

4.3.2 Beziehungsrelationen (zunächst)

Alle Primärschlüssel der beteiligten Entitätsrelationen werden zusammengesetzter Primärschlüssel der Beziehungsrelation. Sonstige Attribute der Beziehungen werden ebenfalls übernommen:

a) AngestellterFährtAuto (nummerschild#, personalNr#)

b) MitarbeiterGehörtZuAbteilung (personalNr#, abteilungsNr#, funktion)

c) IstVorgesetzterVon (vorgesetzterNr#, personalNr#)
Da beide Primärschlüssel der beteiligten Entitätsrelationen gleich heißen, wird umbenannt: die personalNr. des Vorgesetzten wird zu vorgesetztenNr.

d) MitarbeiterArbeitetAnProjekt (personalNr#, projektId#, prozAnteil, tätigkeit)

e) ProjektHatBericht (projektId#, fortlaufendeNr#)

4.3.3 Vereinfachung der Schlüssel bei den Beziehungsrelationen

Außer der n:m-Beziehung in d) MitarbeiterArbeitetAnProjekt, können die Primärschlüssel der anderen Beziehungsrelationen vereinfacht werden:

i. AngestellterFährtAuto (nummerschild#, personalNr#)
Bei einer 1:1-Beziehung kann man frei wählen. Allerdings wird hier nummerschild als Primärschlüssel gewählt, um beim späteren Verschmelzen keine null-Werte zu erhalten. Denn jeder Firmenwagen hat immer einen zugehören Angestellten, der ihn fährt. Im umgekehrten Fall gilt dies nicht, da nicht jeder Angestellte einen Firmenwagen hat.

ii. MitarbeiterGehörtZuAbteilung (personalNr#, abteilungsNr#, funktion)
Bei einer 1:n-Beziehung muss der n-Teil der Primärschlüssel werden, da hieraus eindeutig die andere Entität (1-Teil) folgt. Dies ist die personalNr.

iii. IstVorgesetzterVon (vorgesetzterNr#, personalNr#)
Wie eben: Die n-Seite ist personalNr, denn hieraus folgt eindeutig der Vorgesetzte.

iv. ProjektHatBericht (projektId#, fortlaufendeNr#)
Dies ist der Sonderfall von schwachen Entitäten: Da der n-Teil bereits der

kombinierte Primärschlüssel aus der starken und der schwachen Entität ist, ändert sich nichts.

4.3.4 Verschmelzen

Hier werden die Beziehungsrelationen mit passenden Entitätsrelationen verschmolzen, sofern dies aus Sicht der entstehen null-Werte sinnvoll ist.

Alle Tabellen i bis iv werden verschmolzen. Im Fall iv ändert sich nichts an der Entitätsrelation.

4.3.5 Finale Tabellen

Hinweis: Aus didaktischen Gründen wird in der Tabelle Mitarbeiter das Feld abteilungsNr umbenannt in abteilungsId (kein natural join mehr möglich).

a) Firmenwagen (nummerschild, marke, modell, personalNr#)

b) Mitarbeiter (personalNr, vorname, nachname, anschrift, gebDatum, gehalt, abteilungsId#, funktion, vorgesetzerNr#)

c) Arbeiter (personalNr, arbeitsplatz)

d) Angestellter (personalNr, telefonNr)

e) Abteilung (abteilungsNr, bezeichnung)

f) Projekt (projektId, bezeichnung)

g) StatusBericht (projektId#, fortlaufendeNr, datum, inhalt)

h) MitarbeiterArbeitetAnProjekt (personalNr#, projektId#, prozAnteil, tätigkeit)

4.4 Überprüfen der Relationen auf Normalformen

4.4.1 Erste Normalform

Alle Tabellen liegen in erster Normalform vor, da alle Attribute nur aus atomaren Datentypen bestehen. Der DB-Entwickler vermutet aber, dass die Anwender später gezielt nach Straße, Hausnr., PLZ und Ort suchen wollen. Deshalb wird das Feld Mitarbeiter.anschrift entsprechend aufgesplittet.

- Mitarbeiter (personalNr, vorname, nachname, strasse, hausNr, plz, ort, gebDatum, abteilungsId#, funktion, vorgesetzerNr#)

4.4.2 Zweite Normalform

Ab der 2NF brauchen wir etwaige funktionale Abhängigkeiten. Diese stellen wir hier auf:

- Firmenwagen

  (1) nummernschild → modell, marke, personalNr

  (2) modell → marke

- Mitarbeiter

  (3) personalNr → vorname, nachname, strasse, hausNr, plz, ort, gebDatum, abteilungsNr, funktion, vorgesetzerNr

  (4) plz → ort

Alle anderen Relationen haben keine weiteren Abhängigkeiten, außer dass alle Attribute vom Schlüssel abhängen. Somit sind auch alle Schlüsselkandidaten bestimmt.

Da es keine zusammengesetzten Schlüssel in den Relationen gibt und dies eine Voraussetzung für eine Verletzung der 2NF ist, sind alle Tabellen in 2NF.

4.4.3 Dritte Normalform

In der 3NF achten wir darauf, dass es keine transitiven Abhängigkeiten eines Nicht-Schlüssel-Attributs zu einem anderen Nicht-Schlüssel-Attribut gibt. Dies ist aufgrund von (2) und (4) aber der Fall, sodass die 3NF verletzt wird. Hier hilft aufsplitten:

Mitarbeiter wird geteilt in

- Ort (<u>plz</u>, ortsname) - neue Relation
- Mitarbeiter2 (personalNr, vorname, nachname, strasse, hausNr, plz#, gebDatum, abteilungsId#, funktion, vorgesetzerNr#)

sowie

- Auto (<u>modell</u>, marke) - neue Relation
- Firmenwagen2 (<u>nummerschild</u>, modell#, personalNr#)

4.4.4 Finale Relationen

Hinweis: Mitarbeiter2 und Firmenwagen2 werden umbenannt.

a) Auto (modell, marke)

b) Firmenwagen (nummernschild, modell#, personalNr#)

c) Ort (plz, ortsname)

d) Mitarbeiter (personalNr, vorname, nachname, strasse, hausNr, plz#, gebDatum, abteilungsId#, funktion, vorgesetzerNr#)

e) Arbeiter (personalNr, arbeitsplatz)

f) Angestellter (personalNr, telefonNr)

g) Abteilung (abteilungsNr, bezeichnung)

h) Projekt (projektId, bezeichnung)

i) StatusBericht (projektId#, fortlaufendeNr, datum, inhalt)

j) MitarbeiterArbeitetAnProjekt (personalNr#, projektId#, prozAnteil, tätigkeit)

4.5 Erstellen der Datenbank

Dieser Abschnitt beschreibt, wie die Datenbanken (relational/objekt-orientiert) angelegt und mit Beispieldaten gefüllt werden.

4.5.1 Erstellen der Tabellen unter MariaDB

Zunächst wird mit Administratorrechten ein neues Datenbankschema `firmenwelt` sowie ein technischer Benutzer `companydemo` angelegt. Hierzu kann das Skript `./sql/create-db.sql` ausgeführt werden.

Anschließend können die Tabellen erzeugt werden. Hierfür kann bereits der neue Benutzer `firmenwelt` oder der Administrator-Benutzer genutzt werden. Das Skript ./`sql/create-tables.sql` legt die Tabellen an und erzeugt einige Beispieldaten.

Abbildung 3 zeigt die entstandenen finalen Tabellen in der Ansicht als EER-Diagramm der MySQL-Workbench. Angaben über die Indizes und referentielle Integrität sind dem SQL-Skript zur Erzeugung der Tabellen zu entnehmen (siehe `./sql/create-tables.sql`).

Das Skript `./sql/create-data.sql` erzeugt Beispieldaten direkt über SQL-DML-Befehle. Wichtig ist, dass die Tabellen zuvor leer sind (am besten durch Neuerzeu-

gen der Tabellen), da sonst Fehlermeldungen durch referentielle Integritätsverletzungen zu erwarten sind.

Alternativ lassen sich die Tabellen auch über ein Java-Programm mit Daten füllen, indem `net.gumbix.dba.companydemo.test.ExampleData` ausgeführt wird und der Datenbankzugriff z.B. auf `data.jdbcLocal()` gesetzt wird. Weitere Informationen sind im Abschnitt 5.5 zu finden.

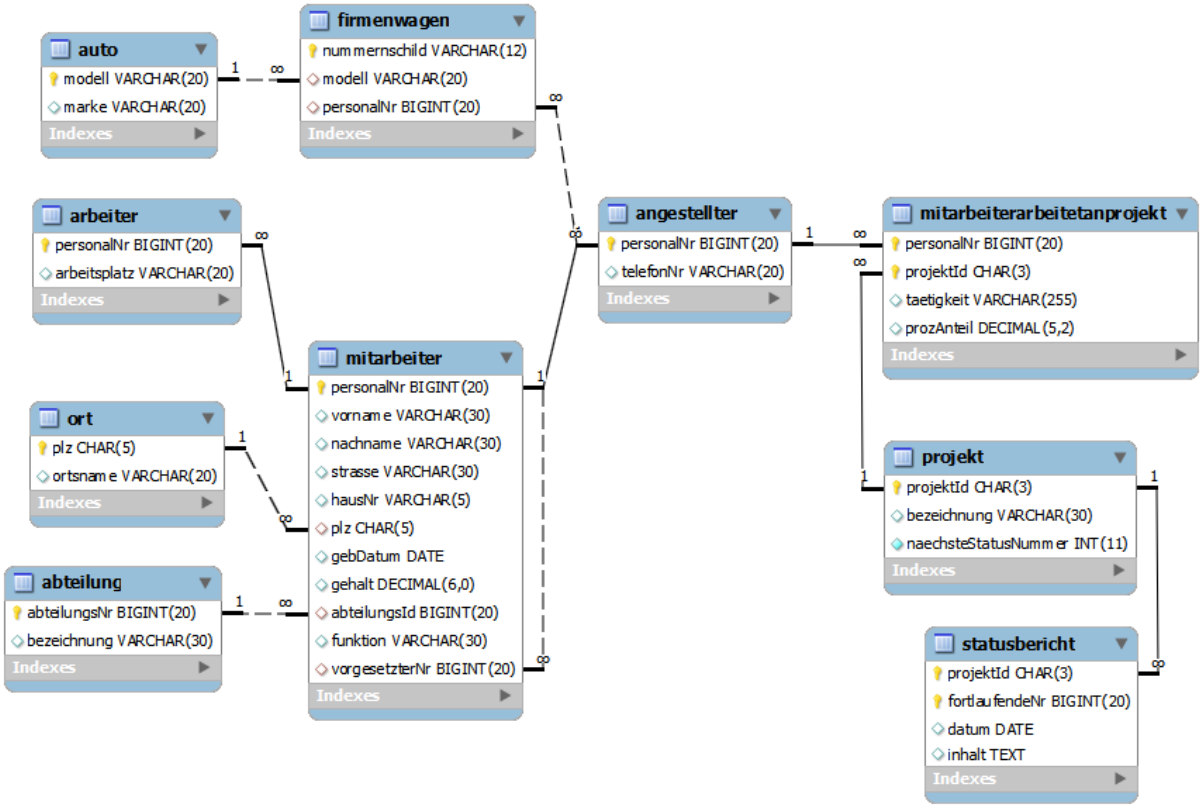


Abbildung 3: Tabellen

4.5.2 Erstellen der Datenbank unter DB4O

Unter DB4O lassen sich die Daten nur über die Programmiersprache erstellen. Hierzu wird das Programm `net.gumbix.dba.companydemo.test.ExampleData` ausgeführt wird und der Datenbankzugriff z.B. auf `data.db4oEmbedded()` gesetzt wird. Weitere Informationen sind im Abschnitt XX zu finden.

4.5.3 Erstellen der Datenbank unter MongoDB

Unter der Voraussetzung, dass MongoDB auf Ihrem Rechner lokal als Server installiert ist (siehe Abschnitt 3.3.4), kann eine dokumenten-orientierte Datenbank mit

MongoDB erstellt werde. Hierzu wird das Programm um die Klasse `net.gumbix.dba.companydemo.MongoDb` erweitert, die eine lokale NoSQL-Datenbank erstellt.

Mit dem das Programm `net.gumbix.dba.companydemo.test.ExampleData` und dem Datenbankzugriff `data.mongoLocal()` lässt sich die Mongo-DB mit Beispieldaten initialisieren.

4.5.4 Erzeugen von Beispieldaten über Java-API

Tbd.

4.5.5 Umfang der Beispieldaten

Diese Anwendung enthält eine Reihe von Beispieldaten, mit der alle wichtigen Konzepte der DB-Programmierung ausprobiert werden können. Das Erzeugen der Beispieldaten ist in den vorigen Abschnitten beschrieben worden. Hier wird erläutert, welchen Umfang die Daten haben. Prinzipiell gilt, dass die Beispieldaten die gleichen sind, egal ob sie über das SQL-Skript oder die Java-API erstellt wurden.

Es gibt drei Fahrzeugtypen und fünf Fahrzeuge, von denen eins ein Pool-Auto ist.

5 Design der Java-/Scala-Anwendung

Dieser Abschnitt behandelt das Design der Anwendungssoftware, d.h. des Java- bzw. Scala-Programms.

5.1 Allgemeine Hinweise

Aus didaktischen Gründen wird auf zahlreiche Methoden und Technologien aus dem Bereich der Softwareentwicklung verzichtet, da diese den Rahmen der Vorlesung sprengen würden. Beispielsweise wird auf Frameworks wie Spring zur Integration der Komponenten verzichtet. Ebenso werden aus Gründen der Einheitlichkeit und Einfachheit Exceptions, die in der Java-/Scala-Anwendung auftreten, generell nach oben durchgereicht und in der UI angezeigt. Außerdem geht es bei dieser Software nicht darum ein Programm mit ausgefeilter Fehlerbehandlung zu erstellen, sondern um den Aspekt der Datenbankanbindung. Es soll bei der Fehlerbehandlung lediglich sichergestellt werden, dass das Programm bei einem Fehler korrekt beendet wird und der Benutzer eine entsprechende Benachrichtigung erhält. Dies erfolgt mit einer einfachen Meldung der Art „Beim Ausführen des Programms ist ein Fehler aufgetreten!“. An diese Nachricht wird die Fehlerbeschreibung angehängt und ausgegeben.

Ferner gelten folgende Konventionen: SQL-Tabellen: alles deutsch, Programmiersprache: alles englisch, auch Kommentare.

5.2 OOD

Das Klassendiagramm der Objekt-Orientierten-Analyse unterscheidet sich von dem in Abschnitt 4.2 (UML-Klassendiagram) entwickelten UML-Klassendiagramm. Folgende Erweiterungen und Änderungen gibt es:

- Konstruktoren werden angegeben.
- Methoden werden angegeben.

Die fortlaufende Nummer eines Statusberichts soll automatisch zugewiesen werden. Ebenso soll eine Personalnummer automatisch zugewiesen werden. Abteilungsnummern und Projektnummer hingegen werden vom Benutzer erfragt, da dies ein sprechender Schlüssel sein soll.

Nähere Beschreibung folgt.

5.3 Domänenklassen

Die Datenhaltung (so genannte Persistenz-Schicht) wird von den eigentlichen Java- bzw. Scala-Klassen getrennt, denn es soll einfach möglich sein (per Schalter), die Datenbank auszutauschen. Alle Domänenklassen befinden sich im Package `net.gumbix.dba.companydemo.domain`. Das Klassenmodell für die benutzten Programmiersprachen wird aus dem Modell der OOA (siehe Abschnitt 4.2 UML-Klassendiagram) abgeleitet.

5.3.1 Java

Etliche Klassen haben ausschließlich Attribute und keine sonstigen fachlichen Methoden (`toString()` u.ä. soll hier ignoriert werden). Aus Gründen, auf die hier nicht näher eingegangen wird, gilt es als guter Programmierstil, Attribute in Java über getter- und setter-Methoden zu setzen (so genannte Java Beans). Diesem Ansatz wird hier gefolgt.

5.3.2 Scala

Beschreibung folgt.

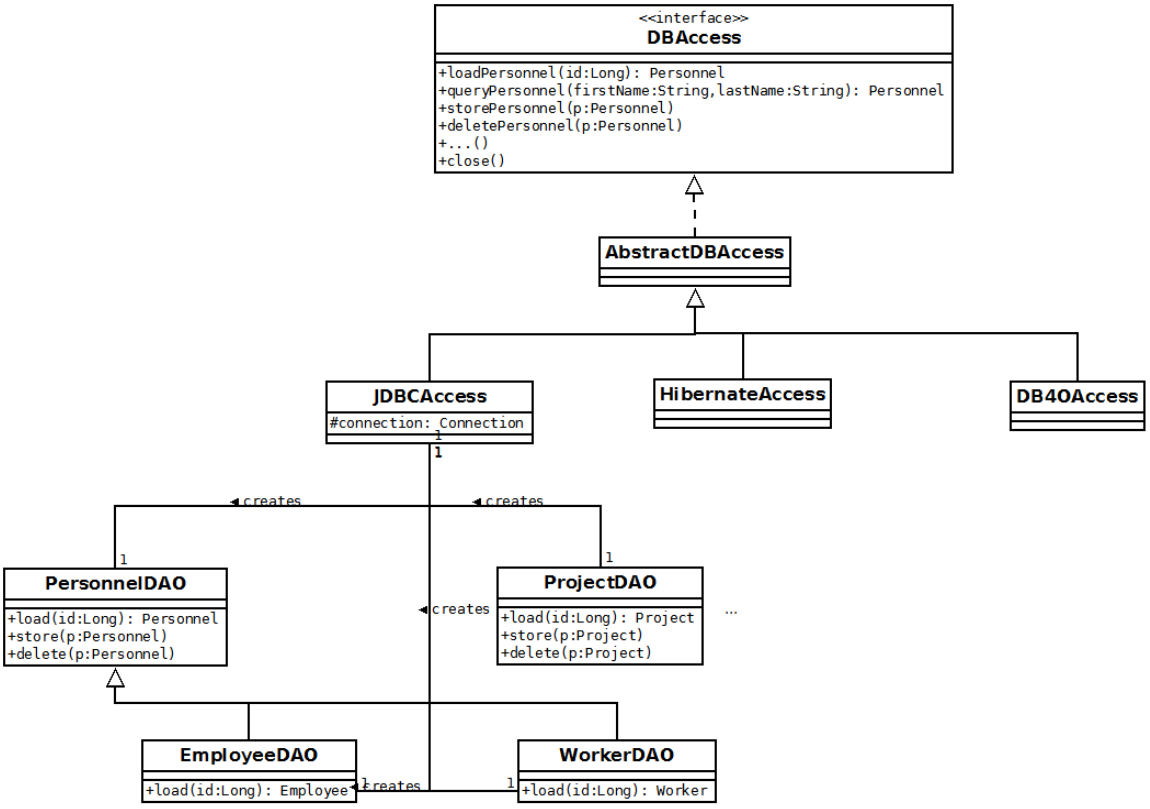
5.4 Zugriff auf eine spezielle Datenbank

Das Interface `net.gumbix.dba.companydemo.db.DBAccess` ist die Schnittstelle zur Persistenzschicht. Hier sind alle Methoden für den Zugriff auf die DB definiert, die die Anwendung benötigt. Typische Methoden haben folgendes Muster:

- `load<Klasse>(id)` lädt ein einzelnes Objekt der Klasse `Klasse` in den Hauptspeicher. Falls es kein Objekt mit dieser ID gibt, wird eine `ObjectNotFoundException` geworfen.
- `query<Klasse>(<Suchkriterium>)` lädt eine Liste von Objekten (leere Liste ist natürlich auch möglich)
- `store<Klasse>(object)` speichert das Objekt in der Datenbank. Existiert es bereits, wird es aktualisiert.
- `delete<Klasse>(object)` löscht das Objekt in der Datenbank. Existiert es bereits, wird es aktualisiert.

Die Implementierung `net.gumbix.dba.companydemo.jdbc.JDBCAccess` realisiert eine JDBC-Umsetzung und `net.gumbix.dba.companydemo.db4o.DB4OAccess` den Zugriff auf die objektorientierte Datenbank. Welche Interface-Implementierung genommen wird, wird per Auswahl im UI in der Klasse `net.gumbix.d-`

`ba.companydemo.application.console.UI` abgefragt. Die abstrakte Klasse `AbstractDBAccess` enthält einige gemeinsame Methoden, die alle Implementierungen benötigen. Dies sind u.a. das Laden von `Employee` und `Worker`-Objekten. Die nachfolgende Abbildung zeigt eine Übersicht.



5.5 Data-Access-Objects

So genannte Data-Access-Objects (DAO) dienen dazu, jeden Zugriff auf die Persistenzschicht (=Datenbank) zu kapseln. Üblicherweise gibt es zu jeder Domänenklasse für jede Zugriffsart genau ein eigenes DAO. Die Zugriffsarten in unserem Fall sind die MySQL-Datenbank und DB4O. Aber auch eine Serialisierung (`java.lang.Serializable`) oder das Speichern in einer XML-Datenbank wären denkbar. Alle DAOs befinden sich im Package `net.gumbix.dba.companydemo.<dbtype>`, wobei `dbtype` für die entsprechende Datenbankzugriffsart steht.

Jedes DAO hat typische Methoden, die analog zu den Methoden in `DBAccess` sind:

- `load(id)` lädt ein einzelnes Objekt in den Hauptspeicher. Falls es kein Objekt mit dieser ID gibt, wird eine `ObjectNotFoundException` geworfen.
- `query(<Suchkriterium>)` lädt eine Liste von Objekten (leere Liste ist natürlich auch möglich)

- `store(object)` speichert das Objekt in der Datenbank. Existiert es bereits, wird es aktualisiert.
- `delete(object)` löscht das Objekt in der Datenbank. Kann es aufgrund von referentieller Integrität nicht gelöscht werden, wird eine Exception geworfen.

Im Falle der relationalen DB befinden sich im DAO meist SQL-Statements.

5.5.1 DAO für JDBC-Zugriff

Beschreibung folgt.

- DB-Anbindung (Connection)
- Resultset
- Prepared Statements

5.5.2 DAO für DB4O-Zugriff

DB4O benötigt keine eigenen Data-Access-Objects, da der Zugriff bereits direkt in den Methoden der Klasse `Db4oAccess` erfolgt.

5.5.3 DAO für Hibernate

Beschreibung folgt.

5.5.4 DAO für Mongo-DB

Beschreibung folgt.

6 Spezielle Abfragen

6.1 Berichte

Untermenu 5 im Hauptmenu der Anwendung beinhaltet eine Reihe von Berichten (im englischen oft Reports). Diese Berichte basieren (meist) auf Abfragen, die die Objekte/Datensätze in der Datenbank filtern.

Beschreibung folgt.

6.2 SQL-Abfragen

6.2.1 Zum Ausprobieren

In der Skript-Datei `./sql/some-queries.sql` sind eine Reihe von SQL-Abfragen zum Ausprobieren zu finden.

6.2.2 Tests in Moodle

In Planung: In jedem Semester wird auch ein unverbindlicher Test in Moodle angeboten, in dem die Fertigkeit SQL-Abfragen zu formulieren überprüft wird.

7 Hinweise für Entwickler und Autoren

Beschreibung folgt.

7.1.1 Features

Nächste wichtige Features:

- Hibernate-Anbindung
- Scala + JDBC
- Scala + DB4O
- Web-Frontend?

7.1.2 DB-Export

Das Skript `./sql/create-data.sql` und die Erzeugung der Beispieldaten über die Java-API `net.gumbix.dba.companydemo.test.ExampleData` lässt sich am besten synchron halten, indem die Daten über die API erzeugt werden und dann aus MySQL exportiert werden. Z.B. geht dies gut über Toad und Tools/Export/Export Wizard.

8 Danksagung

An diesem Dokument und am CompanyDemo-Projekt haben folgende Personen mitgewirkt. Vielen Dank an alle!

Name	E-Mail	Zeitraum	Was?
Markus Gumbel	m.gumbel@hs-mannheim.de	permanent	Alles und Nichts
Eyüp Ünlü und Mike Möck		Wintersemester '18	Erste Version der Mongo-DB-Anbindung
Marius Czardybon	m.czardybon@googlemain.com	Wintersemester ‘10/11	JDBC- und DB4O-Implementierung
Patrick Sturm	patrick-sturm@gmx.net	Sommersemester ‘10	User-Interface und Beispieldaten
Sven Haag	sven-haag@gmx.de	Sommersemester ‘10	User-Interface und Beispieldaten