Karlsruher Institut für Technologie

Entwicklung eines Expertensystems für die Planung kerntechnischer Rückbauprojekte

Felix Hübner, Sven Möller, Frank Schultmann

No. 28 | April 2018

WORKING PAPER SERIES IN PRODUCTION AND ENERGY

Entwicklung eines Expertensystems für die Planung kerntechnischer Rückbauprojekte

Felix Hübner, Sven Möller, Frank Schultmann

Chair of Business Administration, Production and Operations Management,
Institute for Industrial Production (IIP),
Karlsruhe Institute of Technology (KIT),
Hertzstr. 16, building 06.33, 76187 Karlsruhe,
Tel.: +49 721 608-44677, email: felix.huebner@kit.edu

Der Rückbau einer kerntechnischen Anlage ist ein Großprojekt mit geschätzten Rückbaukosten von mehreren hundert Millionen Euro und einer Projektlaufzeit von mehr als zehn Jahren. Aufgrund von politischen, technischen oder ökonomischen Gründen nimmt die Anzahl an rückzubauenden kerntechnischen Anlagen stetig zu. Während der Planung und Durchführung kerntechnischer Rückbauprojekte sind vielfältige Bedingungen und Vorgaben zu beachten. Des Weiteren sind die aufgrund weniger Erfahrungswerte bestehenden Unsicherheiten zu berücksichtigen.

Um diese Unsicherheiten so gut wie möglich zu reduzieren, sollten die bereits gemachten Erfahrungen für die Planung genutzt sowie die noch kommenden Erfahrungen im Rückbau kerntechnischer Anlagen bestmöglich dokumentiert werden. Auf der Basis dokumentierter Erfahrungen können zukünftige Rückbauprojekte deutlich genauer und effizienter geplant werden. Das Ziel dieser Studie besteht in der Entwicklung eines Expertensystems für den Rückbau kerntechnischer Anlagen.

Zunächst wird eine Erfahrungsdatenbank entwickelt, die Teil des Expertensystems ist.

In der Erfahrungsdatenbank können Erfahrungswerte dokumentiert werden. Auf Basis einer Anforderungsanalyse wird ein konzeptionelles Datenmodell erstellt, welches nach der Auswahl des relationalen Datenbankmodells als Ziel-Datenbankmanagement-system schließlich in ein auf dieses abgestimmte logische Datenbankdesign umgewandelt wird. Das logische Datenbankdesign wird in Microsoft Access implementiert.

Für ein neu zu planendes kerntechnisches Rückbauprojekt soll durch gezielte Abfragen auf die in der Erfahrungsdatenbank gespeicherten Erfahrungswerte zugegriffen werden. Hierzu werden im Expertensystem regelbasierte Abfragen erstellt, mit denen relevante Erfahrungswerte für ein neu zu planendes kerntechnisches Rückbauprojekt aus der Erfahrungsdatenbank gefiltert und exportiert werden können.

Entwicklung eines Expertensystems für die Planung kerntechnischer Rückbauprojekte

Felix Hübner • Sven Möller • Frank Schultmann

Karlsruhe Institute of Technology (KIT),
Institute for Industrial Production (IIP)
Hertzstraße 16,
D-76187 Karlsruhe, Germany

felix.huebner@kit.edu

frank.schultmann@kit.edu

Vorbemerkung:

Die vorliegende Studie wurde im Rahmen der Masterarbeit von Herrn Sven Möller in Zusammenarbeit mit seinem Betreuer Herrn Felix Hübner im Jahr 2017 erarbeitet.

Kurzfassung

Der Rückbau einer kerntechnischen Anlage ist ein Großprojekt mit geschätzten Rückbaukosten von mehreren hundert Millionen Euro und einer Projektlaufzeit von mehr als zehn Jahren. Aufgrund von politischen, technischen oder ökonomischen Gründen nimmt die Anzahl an rückzubauenden kerntechnischen Anlagen stetig zu. Während der Planung und Durchführung kerntechnischer Rückbauprojekte sind vielfältige Bedingungen und Vorgaben zu beachten. Des Weiteren sind die aufgrund weniger Erfahrungswerte bestehenden Unsicherheiten zu berücksichtigen.

Um diese Unsicherheiten so gut wie möglich zu reduzieren, sollten die bereits gemachten Erfahrungen für die Planung genutzt sowie die noch kommenden Erfahrungen im Rückbau kerntechnischer Anlagen bestmöglich dokumentiert werden. Auf der Basis dokumentierter Erfahrungen können zukünftige Rückbauprojekte deutlich genauer und effizienter geplant werden. Das Ziel dieser Studie besteht in der Entwicklung eines Expertensystems für den Rückbau kerntechnischer Anlagen.

Zunächst wird eine Erfahrungsdatenbank entwickelt, die Teil des Expertensystems ist.

In der Erfahrungsdatenbank können Erfahrungswerte dokumentiert werden. Auf Basis einer Anforderungsanalyse wird ein konzeptionelles Datenmodell erstellt, welches nach der Auswahl des relationalen Datenbankmodells als Ziel-Datenbankmanagementsystem schließlich in ein auf dieses abgestimmte logische Datenbankdesign umgewandelt wird. Das logische Datenbankdesign wird in Microsoft Access implementiert.

Für ein neu zu planendes kerntechnisches Rückbauprojekt soll durch gezielte Abfragen auf die in der Erfahrungsdatenbank gespeicherten Erfahrungswerte zugegriffen werden. Hierzu werden im Expertensystem regelbasierte Abfragen erstellt, mit denen relevante Erfahrungswerte für ein neu zu planendes kerntechnisches Rückbauprojekt aus der Erfahrungsdatenbank gefiltert und exportiert werden können.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Bei der Durchführung von Großprojekten ist es keine Seltenheit, dass sich die tatsächlich realisierten Kosten sowie die Dauer fundamental von den geplanten Werten unterscheiden. Beispiele von Großprojekten, bei denen es zu einer beachtlichen Abweichung von den Planwerten gekommen ist, sind nicht schwer zu finden. Beispielhaft können hierfür der Umbau des Stuttgarter Hauptbahnhofs, auch bekannt als Stuttgart 21, sowie der Bau des Flughafens Berlin-Brandenburg genannt werden (Beeger, 2015; Südwestrundfunk, 2016). Die bei der Durchführung realisierten enormen Abweichungen von den geplanten Werten lassen auf eine ausgeprägte Unsicherheit bei der Planung von solchen Großprojekten schließen.

Auch der Rückbau kerntechnischer Anlagen fällt mit einer geschätzten Projektdauer von mehr als zehn Jahren und Kosten von derzeit bis zu mehr als einer Milliarde Euro je Anlage in die Kategorie eines Großprojekts. Beispielsweise hat der Rückbau des deutschen Kernkraftwerks Würgassen vom Antrag auf Stilllegung im Jahre 1995 bis zum erfolgreichen Abschluss der Rückbauarbeiten im Jahr 2014 fast 20 Jahre gedauert (E.ON Kernkraft GmbH, 2014, S. 6ff.). Auch mit den letztendlich realisierten Kosten von rund einer Milliarde Euro konnten die zu Projektbeginn veranschlagten Kosten von ca. 500 Millionen Euro nicht eingehalten werden (Heide, 2015).

Da der Rückbau von kerntechnischen Anlagen nicht nur Einzelfälle umfasst, sondern in den nächsten Jahrzehnten weltweit eine Vielzahl von kerntechnischen Anlagen rückgebaut werden (vgl. u.a. Hübner et al., 2017), besteht das Interesse der Betreiber und Rückbauunternehmen darin, die Planung solcher Projekte robuster zu machen, sodass Kosten- und Zeitüberschreitungen dieser Größenordnungen vermieden werden. Um einen robusten und kostenminimalen Plan unter der Berücksichtigung von Unsicherheiten bestimmen zu können, sind gemäß Hübner et al. (2016) zwei Elemente von Bedeutung. Zum einen die Dokumentation von Erfahrungswerten, um Best Practices ermitteln zu können. Zum anderen eine Methode, die aus den Inputdaten der Erfahrungswerte einen durchführbaren und kostenminimalen Projektplan unter der Berücksichtigung von Unsicherheiten bestimmt.

Diese Studie beschäftigt sich mit Ersterem: der Dokumentation und Bereitstellung von Erfahrungswerten im Rückbau von kerntechnischen Anlagen. Dazu wird ein Expertensystem entwickelt, in dem gesammelte Erfahrungen gespeichert werden und

durch regelbasierte Abfragen für zukünftige Projektplanungen beim Rückbau kerntechnischer Anlagen abgefragt werden können.

Beginnend wird in Kapitel 2 die Grundlage für die anstehende Datenbankentwicklung gelegt. Dazu werden verschiedene mögliche Varianten von Entwicklungsprozessen mitsamt ihren spezifischen Phasen näher beleuchtet und miteinander verglichen. Als Ergebnis dieses Prozessvergleichs wird ein resultierender Datenbankentwicklungsprozess herausgearbeitet, anhand dessen die Entwicklung der Erfahrungsdatenbank im Rahmen dieser Studie vorgenommen wird.

Kapitel 3 behandelt die erste Phase des aus Kapitel 2 gewonnenen Entwicklungsprozesses: die Anforderungsanalyse.

Auf der Basis der Anforderungsanalyse folgt in Kapitel 4 die Erstellung eines konzeptionellen Datenmodells. Dabei wird aus dem in Kapitel 3 gewonnenen Anforderungskatalog ein plattformunabhängiges Datenschema entwickelt. Dieses gibt den zu speichernden Daten Struktur und dient als Basis für die plattformabhängige Entwicklung einer Datenbank.

Anschließend wird in Kapitel 5 die Plattform ausgewählt, auf der die Erfahrungsdatenbank für den Rückbau kerntechnischer Anlagen letztendlich implementiert werden soll. Dies umfasst die Auswahl des Datenbankmodells sowie einer passenden Software, mit deren Hilfe die Datenbank implementiert und betrieben wird.

Das konzeptionelle Datenmodell aus Kapitel 4 wird in Kapitel 6 in ein logisches Datenschema überführt. Dabei findet eine Anpassung des konzeptionellen Datenmodells an das in Kapitel 5 ausgewählte Datenbankmodell statt.

Um den Entwicklungsprozess abzuschließen, wird in Kapitel 7 auf den physikalischen Entwurf der zu entwickelnden Datenbank eingegangen.

Mit Hilfe des geschilderten Vorgehens wird schließlich eine Datenbank implementiert, die den Anforderungen des Rückbaus kerntechnischer Anlagen gerecht wird. Deren Umrisse werden in Kapitel 8 vor allen Dingen anhand der Handhabung der Dateneingabe und Datenänderung sowie der Datenabfrage vorgestellt.

Zuletzt wird die Studie in Kapitel 9 zusammengefasst, kritisch gewürdigt und es wird ein Ausblick auf zukünftige Arbeiten gegeben.

2 Projektablauf der Datenbankentwicklung

Um sicherstellen zu können, dass eine Datenbank über einen längeren Zeitraum hinweg den an sie gestellten Anforderungen und Erwartungen gerecht wird, ist ein gut durchdachter Datenbankentwurf unumgänglich. Da die abzubildende Realität mitunter sehr komplex sein kann, ist dem Datenbankentwurf als Prozess der Überführung der Realität in eine modellhafte Abbildung besonderes Augenmerk zu schenken. In dieser Phase werden die Grundbausteine gesetzt, sodass viele Fehler und Schwachstellen bereits im Voraus ausgeschlossen werden können und ein qualitativ hochwertiges Endprodukt sichergestellt werden kann.

Das wohl bekannteste Modell zur Entwicklung von Software ist das Wasserfallmodell. Es beschreibt neben den beiden Schritten der Analyse und dem Schreiben des Programmcodes, noch weitere Schritte (vgl. Abbildung 1). Das Durchlaufen der Schritte geschieht aufgrund zahlreicher Änderungen, die sich im Laufe des Entwicklungsprozesses ergeben, und Anpassungen, die aufgrund von Fehlern vorgenommen werden müssen, nicht in streng sequentieller Abfolge, sondern iterativ (Royce, 1987, S. 328ff.). Beispielsweise kann ein Fehler, der beim Testen der Software auftritt, ein Umschreiben des Programmcodes nach sich ziehen oder auch eine Anpassung des Programm-Designs zur Folge haben. Dies erzwingt wiederum eine Anpassung im Code und somit muss zwei oder sogar noch mehr Phasen zurück gesprungen werden. Gegebenenfalls müssen einzelne Schritte auch mehrfach wiederholt werden.

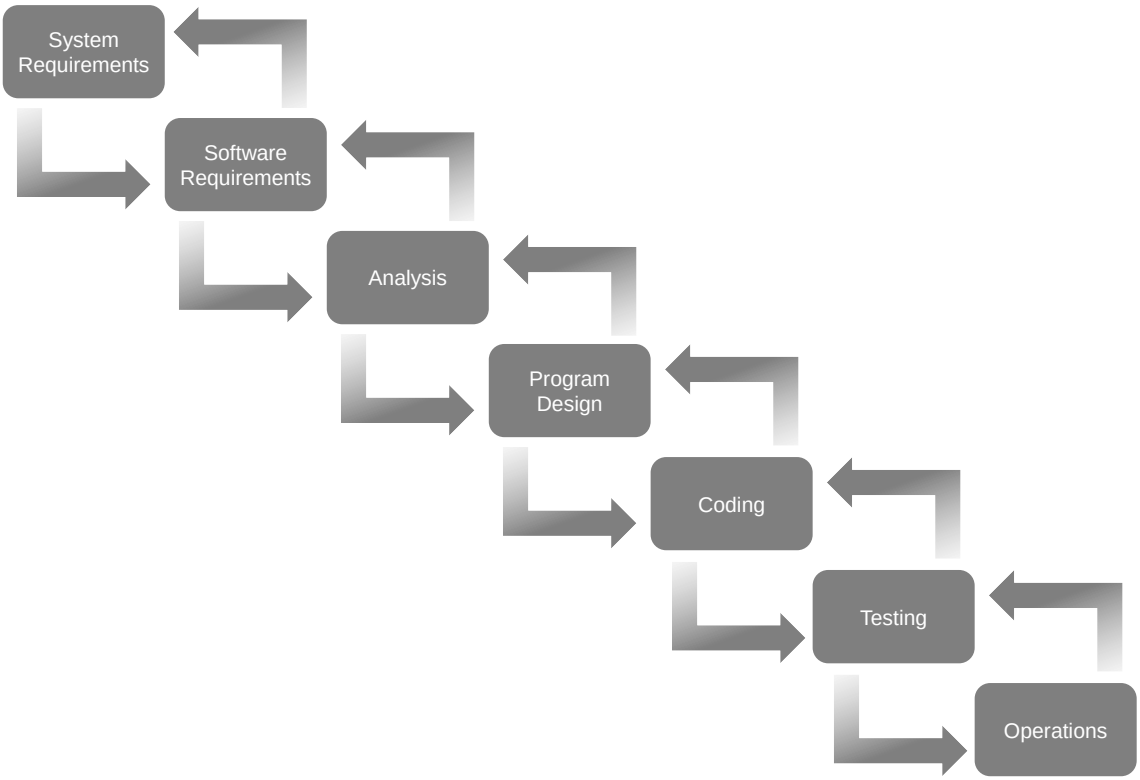


Abbildung 1: Wasserfallmodell der Softwareentwicklung (Eigene Darstellung in Anlehnung an Royce, 1987, S. 330)

Ausgehend von diesem allgemeinen Modell zur Softwareentwicklung haben einige Autoren Anpassungen des Wasserfallmodells an die Datenbankentwicklung unternommen, um den Datenbankentwurf mit höchstmöglicher Wahrscheinlichkeit zu einem erfolgreichen Ergebnis zu führen. Die Datenbankentwurfsprozesse verschiedener Autoren werden im Folgenden vorgestellt.

2.1 Datenbankentwurfsprozess nach Geisler

Geisler (2014) hat den Datenbankdesign-Prozess in vier Schritte unterteilt, die seiner Meinung nach die vielversprechendste Vorgehensweise darstellen:

1. Erarbeiten eines konzeptionellen Designs
2. Auswahl eines geeigneten Datenbankmanagementsystems (DBMS)
3. Erarbeiten des logischen Designs
4. Physikalisches Design der Datenbank

Obwohl die vier Phasen grundsätzlich aufeinanderfolgend und aufeinander aufbauend sind, ist der Datenbankdesign-Prozess nach Geisler (2014) vielmehr als iterativer Prozess zu verstehen (Geisler, 2014, S. 295f.).

Mit dem konzeptionellen Modell wird ein Datenmodell erstellt, das die Realität möglichst genau abbilden soll. Grundlage hierzu liefert die Analyse der realen Umgebungsbedingungen und der Anforderungen, die sich aus der zu lösenden Aufgabe ableiten lassen. Dabei wird jedoch die eigentliche Realisierung einer Datenbank außen vor gelassen, d.h. es ist hier noch unerheblich, welche Art von Datenbank und welches DBMS später genutzt werden. Dadurch bleibt das konzeptionelle Datenmodell plattformunabhängig (Geisler, 2014, S. 296).

Der nächste notwendige Schritt gemäß Geisler (2014) ist die Auswahl eines für die Implementierung des konzeptionellen Modells geeigneten DBMS. Es gibt verschiedenste DBMS, die Datenbanken verschiedenster Arten bearbeitbar machen. Neben den spezifischen Anforderungen, die ein DBMS im Einzelfall erfüllen muss, ist vor allem der Kostenfaktor von Interesse. Anschaffungs- bzw. Lizenzkosten, Kosten für Betrieb und Installation sowie auch die Ausbildung der Mitarbeiter für den Umgang mit dem DBMS sind hierbei ausschlaggebend. Weitere Kriterien zur Auswahl eines DBMS sind die Plattformabhängigkeit bzw. Portierbarkeit, die Hardwareanforderungen und die Kompatibilität zu bereits bestehenden Systemen des DBMS (Geisler, 2014, S. 305).

Anschließend an die Auswahl des DBMS folgt das logische Design des Datenmodells. Hierbei wird das plattformunabhängige konzeptionelle Modell in eine von dem ausgewählten DBMS abhängige Zielplattform überführt (Geisler, 2014, S. 306). Beispielsweise kann das konzeptionelle Datenmodell in ein relationales Datenmodell überführt werden, welches mit Hilfe von relationalen Datenbankmanagementsystemen (RDBMS) verwaltet werden kann.

Als vierten und letzten Schritt im Entwurfsprozess gibt Geisler (2014) das physikalische Design einer Datenbank an. Hierbei geht es darum, sich mit der physikalischen Speicherung der Datenelemente zu befassen, um die Performance bestmöglich auszunutzen. Dies beinhaltet unter anderem die Festlegung der Speicherorte der einzelnen Elemente der Datenbank sowie den Zugriff darauf. Das physikalische Design ist maßgeblich abhängig von der zur Verfügung stehenden Hardware und dem verwendeten DBMS (Geisler, 2014, S. 306).

2.2 Datenbankentwurfsprozess nach Lang und Lockemann

Einen ähnlichen Entwurfsprozess wie Geisler (2014) beschreiben Lang und Lockemann (1995). Der Entwurfsprozess von Lang und Lockemann (1995) umfasst ebenfalls vier Phasen.

1. Anforderungsanalyse
2. Konzeptioneller Entwurf
3. Logischer Entwurf
4. Physischer Entwurf

Auch dieser Phasenablauf ist nicht streng sequentiell, sondern als iterativ verzahnter Prozess zu verstehen (Lang und Lockemann, 1995, S. 291ff.).

Als neuer eigenständiger Schritt kommt im Vergleich zu Geisler die Anforderungsanalyse hinzu, die somit von Lang und Lockemann (1995) stärker gewichtet wird. Die Auswahl eines geeigneten DBMS findet sich in diesem Modell jedoch nicht als eigenständige Phase wieder.

In der hinzugekommenen Anforderungsanalyse wird der Anwendungsbereich abgegrenzt sowie Anforderungen an die Datenbank herausgearbeitet und Ziele definiert. Dazu wird die reale Welt als Ausgangsbasis genommen. Anfangs vage Vorstellungen und Aussagen dieser Ausgangsbasis werden zunächst konkretisiert, um sie analysieren und daraus Ergebnisse ableiten zu können (Lang und Lockemann, 1995, S. 293).

2.3 Datenbankentwurfsprozess nach Saake, Sattler und Heuer

Das Phasenmodell zur Datenbankentwicklung von Saake et al. (2013) umfasst mehr als vier Phasen. Um eine Datenbank zu entwerfen, wird ein Prozess aus sieben Schritten als Handlungsempfehlung vorgeschlagen.

1. Anforderungsanalyse
2. Konzeptioneller Entwurf
3. Verteilungsentwurf
4. Logischen Entwurf
5. Datendefinition
6. Physischer Entwurf
7. Implementierung und Wartung

Auch hierbei ist keine starre sequentielle Abfolge der verschiedenen Phasen vorgesehen, sondern Rückkopplungen und Iterationen durchaus erwünscht und notwendig (Saake et al., 2013, S. 122f.).

Auch in diesem Modell gibt es Phasen, die in den beiden vorhergehend beschriebenen Modellen nicht als eigenständige Phase genannt wurden. Dies sind die Phasen Verteilungsentwurf, Datendefinition sowie Implementierung und Wartung.

Sofern die Daten auf mehreren verschiedenen Datenträgern, wie beispielweise Rechnern oder Servern, verteilt gespeichert werden, muss die Art und Weise der Verteilung festgelegt werden. Der Verteilungsentwurf kann dabei unterschiedliche Objekttypen auf unterschiedlichen Knoten vorsehen. Hierbei stellt die Gestaltung der Beziehungen der Objekttypen untereinander eine Herausforderung dar. Alternativ kann beispielweise eine vertikale oder horizontale Fragmentierung einzelner Objekttypen stattfinden. Bei einer horizontalen Verteilung liegen verschiedene Ausprägungen eines Objekttyps auf unterschiedlichen Knoten. Bei einer vertikalen Verteilung liegen einzelne Attribute von Objekttypen verteilt vor (Saake et al., 2013, S. 127f.).

Die Datendefinition ist die direkt auf den logischen Entwurf folgende Phase. Hierbei wird das logische Schema, das ein Datenbankschema für das gewünschte Zieldatenbankmodell darstellt, in das konkrete Schema des implementierten DBMS umgesetzt (Saake et al., 2013, S. 129). Ist beispielsweise ein relationales Datenschema als logischer Entwurf vorhanden, dann wäre eine mögliche Datendefinition eine Umsetzung passend zum RDBMS Microsoft Access, Microsoft SQL Server oder auch MySQL.

Die letzte Phase im Phasenmodell von Saake et al. (2013) ist die Implementierung und Wartung. Das aus dem Entwicklungsprozess resultierende Datenbanksystem ist keinesfalls unveränderlich. Mit der Zeit müssen die Wartung von bestehenden Funktionen, die Anpassungen an etwaige neue Anforderungen und die Optimierungen stetig vorgenommen werden. Auch eine etwaige Portierung der Datenbank auf ein anderes DBMS fällt in diese Phase (Saake et al., 2013, S. 130).

2.4 Resultierender Entwicklungsprozess

Die in den vorangegangenen Abschnitten erläuterten Datenbankentwicklungsprozesse haben viele Gemeinsamkeiten, weichen aber in einzelnen Punkten voneinander ab und setzen somit unterschiedliche Schwerpunkte. Um die im Rahmen dieser Studie gestellten Problemstellungen zielführend bearbeiten zu können, wird im weiteren Verlauf dieser Studie ein resultierender Entwicklungsprozess verwendet, der auf den Grundkonzepten aller drei vorgestellten Entwurfsprozesse aufbaut und an die vorliegende Aufgabenstellung angepasst ist. Dieser ist in Abbildung 2 dargestellt. Um die Vorteile der in den Kapiteln 2.1, 2.2 und 2.3 vorgestellten Entwurfsprozessen zu

nutzen, wird ein eigener Entwurfsprozess entwickelt. Die Anwendung eines der vorgestellten Entwurfsprozesse würde die Vorteile anderer Entwurfsprozesse außen vor lassen, was vermieden werden soll. Im Folgenden wird das Zustandekommen des resultierenden Entwicklungsprozesses dargelegt.

Abbildung 2: Datenbankentwurfsmodell dieser Studie

Alle vorgestellten Phasenmodelle haben gemeinsam, dass sie mit einer Anforderungsanalyse beginnen, um Problemstellung, Aufgabengebiet sowie abzubildende Daten bzw. Datenstrukturen vollständig zu umfassen und zu begreifen (in Geisler (2014) ist die Anforderungsanalyse zwar nicht explizit als separater Schritt aufgeführt, allerdings ist diese im ersten Schritt enthalten). Dies ist unabdingbar, um den auf der Anforderungsanalyse aufbauenden konzeptionellen Datenentwurf gewissenhaft und korrekt durchführen zu können. Die in der Realität bestehenden Zusammenhänge zwischen den zu erfassenden Daten sollen möglichst genau im konzeptionellen Entwurf abgebildet werden.

Auch den konzeptionellen Entwurf haben die drei vorgestellten Phasenmodelle gemeinsam. In dieser Phase werden die aus der Anforderungsanalyse gewonnenen Ergebnisse in ein plattformunabhängiges Datenschema mit allen relevanten Daten und deren Beziehungen untereinander übertragen.

Um dieses plattformunabhängige Datenschema jedoch auf die letztendliche Zielplattform übertragen zu können, muss zunächst die passende Zielplattform gewählt werden. Deswegen wird die zweite Phase aus Geislers Modell (die Auswahl des DBMS) auch in den Prozess für die Entwicklung der Datenbank im Rahmen dieser

Studie übernommen. In dieser Phase ist neben den bereits in Abschnitt 2.1 beschriebenen Kriterien vor allen Dingen zunächst die Auswahl des zu implementierenden Datenbankmodells relevant. Da die meisten DBMS darauf ausgelegt sind, eine Datenbank von nur einem bestimmten Typ zu verwalten, schränkt dies die Auswahl an geeigneten DBMS bereits sehr ein.

Nach der Auswahl eines für die Implementierung der Datenbank geeigneten DBMS muss der plattformunabhängige konzeptionelle Entwurf in eine Zielplattform überführt werden, für die das in der vorhergehenden Phase ausgewählte DBMS zugeschnitten ist. Durch diesen Umstand ist das logische Design abhängig von der ausgewählten Zielplattform sowie vom DBMS und soll an diese optimal angepasst werden. Dadurch bleibt eine möglichst genaue Realitätsabbildung auch im logischen Schema gewährleistet. Gleichzeitig kann das DBMS reibungslos auf die Werte der Datenbank zugreifen und sicher und effizient mit ihnen arbeiten.

Die letzte Phase im resultierenden Prozess ist der physische Entwurf. Wie bereits in Kapitel 2.1 beschrieben wurde, geht es bei diesem darum, Speicherorte für die einzelnen Datenbankelemente zu bestimmen. Über ein intelligent gewähltes physikalisches Design der Datenbank kann die Performance des Systems positiv beeinflusst werden. Besonderes Augenmerk kommt dem physischen Entwurf im Falle einer verteilten Datenbank zu, da hierbei zusätzliche Kommunikationswege über das Netzwerk berücksichtigt werden müssen (Geisler, 2014, S. 306f.).

Im weiteren Verlauf dieser Studie wird anhand des vorgestellten Datenbankentwurfsmodells eine Datenbank für die Sammlung und Abfrage von Erfahrungswerten, das sogenannte Expertensystem, erstellt. Hierbei ist die Datenbank zur Sammlung von Erfahrungswerten (im Folgenden auch als Erfahrungsdatenbank bezeichnet) Teil des Expertensystems. Das Expertensystem zeichnet sich insbesondere durch intelligente Abfragen der in der Erfahrungsdatenbank gespeicherten Daten aus.

3 Anforderungsanalyse

In diesem Kapitel werden, wie in Kapitel 2 von zahlreichen Autoren beschrieben wurde, die realen Begebenheiten des Anwendungsfalls betrachtet, um daraus Anforderungen an eine zu entwickelnde Datenbank abzuleiten. Dies reicht von den festzuhaltenden Inhalten, d.h. was gespeichert werden soll, bis hin zu deren Verarbeitung, was mit den zu speichernden Daten passiert und der schlussendlichen Aufbereitung, beispielweise der Ausgabe der Daten. Im Rahmen dieser Studie soll die Datenbank primär dazu dienen, bestehende Erfahrungen beim Rückbau kerntechnischer Anlagen zu speichern, sodass aus diesen in Zukunft mit Hilfe von regelbasierten Abfragen für eine neue Planung relevante Daten identifiziert werden können.

Um Anforderungen an die zu speichernden Elemente ableiten zu können, muss zuerst ein Überblick über den allgemeinen Ablauf eines Rückbauprojekts gewonnen werden. Der Rückbau einer kerntechnischen Anlage läuft in mehreren Schritten ab.

Anlagenkomponenten, die für den Restbetrieb nicht mehr benötigt werden, werden abgebaut und die bestrahlten Brennelemente werden frühestmöglich aus der Anlage entfernt. In der Regel werden radioaktive Anlagenkomponenten, wie beispielweise der Reaktordruckbehälter und das biologische Schild, im zweiten Schritt abgebaut. Gebäude werden dekontaminiert und die gesamte Anlage wird aus der atom- und strahlenschutzrechtlichen Überwachung entlassen. Verbliebene Gebäude werden entweder weiterverwendet oder konventionell abgerissen.

Anhand dieses schrittweisen Ablaufs der Rückbauarbeiten ist zu erkennen, dass zwischen vielen Rückbauvorgängen zeitliche und/oder technologische Abhängigkeiten bestehen, die eingehalten werden müssen. Dies lässt sich gut mit Methoden des Projektmanagements modellieren. Dabei werden beispielweise mittels einer Work Breakdown Structure (WBS) größere Arbeitspakete auf kleinere Arbeitsschritte heruntergebrochen, um die Komplexität des Projekts beherrschbar zu machen. Auf diesem detaillierteren Level werden auch die zeitlichen Anordnungsbeziehungen zwischen einzelnen Vorgängen festgehalten und Kosten, Dauern sowie Ressourcenbeanspruchungen können für kleine Einheiten besser geschätzt werden (Vanhoucke, 2013, S. 12f.). Besonders im Kontext des Rückbaus kerntechnischer Anlagen muss das Planungsverfahren diese zeitlichen Abhängigkeiten berücksichtigen, sodass Aufgaben in stärker belastetem Umfeld schnell und vor allem unterbrechungsfrei ausgeführt werden können. Dadurch sollen Arbeiter während dieser

Arbeitsschritte nur über einen möglichst kurzen Zeitraum radioaktiver Strahlung ausgesetzt werden.

Diese aus größeren Arbeitspaketen heruntergebrochenen Vorgänge bilden demnach das Kernstück der zu speichernden Daten. Von Interesse sind hierbei vor allen Dingen Daten, die zur besseren Planbarkeit beitragen, wie beispielsweise zeitliche Abhängigkeiten zu anderen Vorgängen. Auch die Bezeichnung der Vorgänge, der Ausführungsort und die Ausführungswahrscheinlichkeit eines Vorgangs sind von Bedeutung. Manche Vorgänge müssen nicht mit Sicherheit durchgeführt werden, sondern deren Notwendigkeit zur Ausführung ergibt sich aus dem Ablauf anderer Vorgänge. Daraus resultiert für manche Vorgänge eine Ausführungswahrscheinlichkeit von unter 100%. Auch sind manche Vorgänge wiederholt durchzuführen, da eine einmalige Durchführung nicht ausreichend ist. Dies kann beispielsweise bei Dekontaminationsvorgängen der Fall sein, wenn die Messung ergibt, dass immer noch zu hohe Strahlungswerte vorliegen. In einem solchen Fall muss der Vorgang so oft wiederholt werden, bis eine Messfreigabe erteilt werden kann oder das Material wird der Endlagerung zugeführt. Daraus leitet sich ab, dass neben der Ausführungswahrscheinlichkeit auch festgehalten werden muss, ob ein Vorgang zyklisch ausführbar ist und wenn ja ob es eine maximale Anzahl an durchführbaren Zyklen gibt und wie hoch diese ist. Auch die vorherrschende radioaktive Strahlung ist eine Information von immenser Bedeutung. Bei hoher radioaktiver Strahlung müssen gegebenenfalls besondere Vorkehrungen getroffen werden, um Mensch und Umwelt nicht zu gefährden bzw. zu beeinträchtigen.

Aus wirtschaftlicher Sicht sind die bei der Ausführung von Vorgängen entstehenden Kosten relevant. Hierbei ist insbesondere die Veränderung der Restbetriebskosten von Interesse. Schreitet der Rückbau voran, dann verkleinert sich im Regelfall die Baustelle, Logistikanforderungen schrumpfen und Kosten zum Betrieb einiger Komponenten verringern sich. Auf diese Weise tragen im Regelfall einzelne durchgeführte Vorgänge zu einer Verminderung der Restbetriebskosten bei.

Des Weiteren können einige Vorgänge in verschiedenen Ausführungsvarianten durchgeführt werden. Eine Wand kann beispielsweise mit einem Hammer oder auch mit einem Bagger eingerissen werden. Diese möglichen sogenannten Ausführungsmodi eines Vorgangs müssen ebenfalls festgehalten werden. Abhängig vom Modus werden unterschiedliche Ressourcen benötigt und der betreffende Vorgang benötigt zur Ausführung womöglich eine unterschiedliche Dauer je nach Modus.

Die unterschiedlichen Modi eines Vorgangs beanspruchen eventuell Ressourcen, die sich in der Art und Menge unterscheiden. Insbesondere ist in der Datenbank festzuhalten, welche Ressourcen und in welchen Mengen ein Modus diese beansprucht. Dazu ist auch eine datenseitige Modellierung der Ressourcen unumgänglich. Hierbei sind neben der Bezeichnung der verwendeten Ressourcen vor allen Dingen die Art der Ressource und die Anschaffungs- sowie die variablen Kosten wichtig für die Projektplanung. Zur Art der Ressource ist wichtig zu wissen, ob es sich um eine erneuerbare Ressource oder eine nicht-erneuerbare Ressource handelt. Erneuerbare Ressourcen stehen unabhängig vom vergangenen Verlauf des Projekts über den gesamten Projektverlauf mit kontinuierlicher Kapazität zur Verfügung. Sie erneuern sich in jeder Zeitperiode und sind wiederholt nutzbar. Gängige Beispiele hierfür sind die menschliche Arbeitskraft oder auch Werkzeuge. Sie werden nicht verbraucht, sondern stehen nach Gebrauch erneut zur Verfügung. Nicht-erneuerbare Ressourcen hingegen stellen ihre Kapazität über den gesamten Projektverlauf zur Verfügung und werden nicht wieder erneuert, sondern werden endgültig verbraucht. Rohmaterialien oder auch Geld stellen Beispiele für nicht-erneuerbare Ressourcen dar (Vanhoucke, 2013, S. 110).

Schlussendlich muss es möglich sein, die bei der Durchführung von Vorgängen bzw. deren Modi gemachten Erfahrungswerte einem Rückbauprojekt zuzuordnen. Neben dem Projektnamen sind hierbei vor allen Dingen der verwendete Reaktortyp sowie dessen Generation von Bedeutung. Es macht beispielweise einen Unterschied, ob es sich um den Rückbau einer Anlage mit einem Siedewasserreaktor oder einem Druckwasserreaktor handelt. Abhängig vom Reaktortyp sind unterschiedliche Bauteile vorhanden, die abgebaut werden müssen. Aus diesem Grund sind unterschiedliche Vorgänge und Modi vonnöten bzw. möglich. Aus der projektplanerischen Sicht sind des Weiteren der Fälligkeitstermin (Deadline) sowie die Restbetriebskosten zu Beginn des Projekts wichtige Daten, die gespeichert werden sollen.

4 Konzeptionelles Datenmodell

Wie bereits in Kapitel 2 beschrieben wurde, ist in der frühen Phase der Entwicklung einer Datenbank die Erstellung eines konzeptionellen Datenmodells unumgänglich. Sämtliche nachfolgenden Entwicklungsschritte bauen auf dem konzeptionellen Datenmodell auf und somit setzen sich etwaige Designfehler im restlichen Entwicklungsprozess fort. Daher ist größtmögliche Sorgfalt und Achtsamkeit bei der konzeptionellen Modellierung geboten (Geisler, 2014, S. 131).

Um einen realen Sachverhalt in einem möglichst exakten und vollständigen Datenmodell grafisch zu beschreiben, eignet sich im Besonderen die Sprache des Entity-Relationship-Modells (ER-Modell). Grundlegende Elemente der Datenmodellierung mit Hilfe des ER-Modells sind aus dem von Chen im Jahr 1976 verfassten Artikels hervorgegangen (Chen, 1976; Jarosch, 2016, S. 27). Seitdem hat es sich „fest im Bereich der Datenbankmodelle etabliert und wird – in abgewandelten Formen – heutzutage faktisch als Standardmodell für frühe Entwurfsphasen der Datenbankentwicklung eingesetzt.“ (Saake et al., 2013, S. 59) Diese breite Durchsetzung in der Praxis des Datenbankentwurfs verdankt das ER-Modell seiner intuitiven Semantik, die eine weniger stark ausgeprägte Formalisierung ausgleicht (Lang und Lockemann, 1995, S. 333). Aufgrund der leichtgewichtigen verwendeten Konzepte und der wenig komplexen Notation ergibt sich zusammen mit der grafischen Komponente des ER-Modells eine Anschaulichkeit und Lesbarkeit und ist somit auch für Nicht-Datenbankspezialisten wie beispielsweise Anwender oder Auftraggeber leicht verständlich. Dies ist insbesondere in der Kommunikation mit diversen Stakeholdern ein erheblicher Vorteil (Saake et al., 2013, S. 51/59).

Bei der konzeptionellen Modellierung mit Hilfe des ER-Modells werden lediglich der Typ und die Struktur der zu modellierenden Daten festgehalten bzw. dargestellt und nicht deren konkreter wertmäßiger Inhalt (Saake et al., 2013, S. 51). Dazu bedient sich das ER-Modell drei grundlegenden Konzepten (Saake et al., 2013, S. 59):

1. Entity (Objekt): Eine zu modellierende Informationseinheit
2. Relationship (Beziehung): Modellierung von Beziehungen zwischen Entities
3. Attribut: Eigenschaft einer Entity oder einer Beziehung

Außerdem erfolgt die Modellierung unabhängig vom logischen Design.

Im Zuge der Entwicklung eines solchen konzeptionellen Datenmodells ist nach Jarosch (2016) in folgenden vier Schritten vorzugehen (Jarosch, 2016, S. 29):

1. **Objektklassifizierung:** Identifizierung von Objekttypen, deren Informationen gespeichert werden sollen.
2. **Abstraktion:** Die Vielfalt aller bekannten Informationen über die Objekttypen wird auf einen definierten Satz relevanter Eigenschaften reduziert.
3. **Identifizierung:** Methode, um Unterscheidungen der verschiedenen Objekte desselben Objekttyps zu ermöglichen.
4. **Sachlogische Zusammenhänge:** Beschreibung der sachlogischen Zusammenhänge der Objekttypen untereinander.

Im Folgenden wird auf diese vier Schritte näher eingegangen. Zunächst werden diese theoretisch beschrieben und in Kapitel 4.6 auf den Anwendungsfalls der Datenbank zur Speicherung von Erfahrungswissen für den Rückbau kerntechnischer Anlagen übertragen. In Kapitel 4.5 werden die ersten drei Normalformen beschrieben, die zur Qualitätssicherung von Datenbanken und zur Qualitätssicherung der Speicherung von Daten verwendet werden.

4.1 Objektklassifizierung

Eine *Entity*, welche auch als *Objekt* bezeichnet wird, ist ein Element, über welches Informationen gespeichert werden. Dies kann eine Person, ein Gegenstand oder etwas Immaterielles, wie beispielsweise ein Prozess oder ein Sachverhalt, sein. Ein *Objekttyp* (oder *entity type*) hingegen ist eine konkrete Klasse von Objekten, die in der Benennung klar voneinander abgegrenzt sind und für dieselben Informationstypen gespeichert werden, die auch in gleicher Art und Weise verarbeitet werden (Jarosch, 2016, S. 31). Es ist also „eine Menge aus gleichartigen Objekten“ (Geisler, 2014, S. 138). Ein Beispiel für einen Objekttyp wäre eine Person. Der Objekttyp ‚Person‘ ist eine Klasse von Objekten, die sich von anderen abgrenzt und für deren Ausprägungen die gleichen Informationen, wie beispielsweise Vorname, Nachname und Geburtsdatum, gespeichert werden. Diese Informationen werden auch *Attribute* genannt (siehe Kapitel 4.2). Es ist jedoch darauf zu achten, dass im konzeptionellen Modell lediglich die Informationstypen und deren Struktur festgehalten werden und nicht deren konkrete wertmäßige Ausgestaltung.

Zur einheitlichen Darstellung von Objekttypen mit Hilfe des ER-Modells werden im Rahmen dieser Studie drei syntaktische Regeln verwendet (Jarosch, 2016, S. 32):

1. Im ER-Modell wird ein Objekttyp durch ein zweigeteiltes Kästchen dargestellt, in dessen Kopfteil der Name des Objekttyps steht.
2. Größe und Position des Kästchens sind unerheblich.
3. Der Name des Objekttyps muss für das gesamte konzeptionelle Datenmodell eindeutig sein und im Singular stehen.

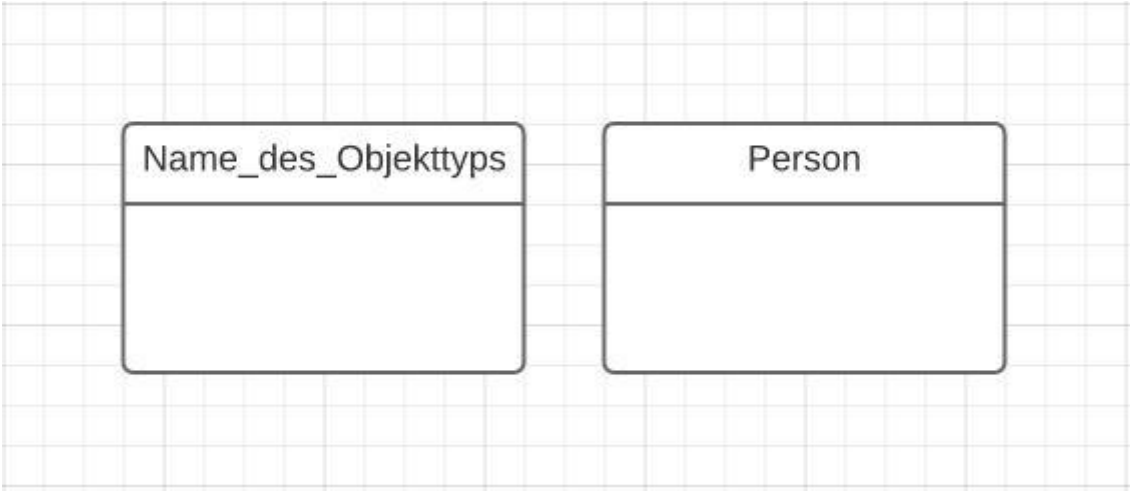


Abbildung 3: Darstellung Objekttyp im ER-Modell

Die grafische Darstellung eines Objekttyps im ER-Modell gemäß den Syntaxregeln ist in Abbildung 3 ersichtlich. Auf der linken Seite ist die allgemeine Darstellung abgebildet und auf der rechten Seite zur Veranschaulichung die des beispielhaften Objekttyps ‚Person'.

Ausgehend von der Anforderungsanalyse aus Kapitel 3 ist im Rahmen der konzeptionellen Modellierung des Anwendungsfalls die Definition von mehreren Objekttypen notwendig.

4.2 Abstraktion auf relevante Eigenschaften

Wie bereits im vorangegangen Abschnitt erläutert wurde, werden Objekte eines Objekttyps in gleicher Weise verarbeitet. Um solche gleichartigen Verarbeitungsprozesse möglich zu machen, müssen über die Objekte eines Objekttyps relevante Eigenschaften gespeichert werden, die Eingabeinformationen benötigen, Ausgabeinformationen liefern oder zur sonstigen Verarbeitung gebraucht werden (Jarosch, 2016, S. 33f.). Die Objekttypen werden neben ihrem Namen durch diese relevanten Eigenschaften, auch *Attribute* genannt, näher beschrieben (Geisler, 2014, S. 139).

Eine mögliche Definition für eine Eigenschaft sowie einen Eigenschaftswert liefert Jarosch (2016) wie folgt:

„Eine *Eigenschaft* (engl. attribute) ist die Benennung für ein relevantes Merkmal aller Objekte, die in einem Objekttyp zusammengefasst werden.“ (Jarosch, 2016, S. 34)

„Ein *Eigenschaftswert* (engl. attribute value) ist eine spezielle Ausprägung, die eine Eigenschaft für ein konkretes Objekt annimmt.“ (Jarosch, 2016, S. 34)

Es ist beim konzeptionellen Modellieren der Daten darauf zu achten, nur die Eigenschaften der Objekttypen zu benennen und zu verwenden und nicht die Eigenschaftswerte, die eine Ausgestaltung eines konkreten Objekts annimmt.

Analog zur Darstellung der Objekttypen werden auch für die Attribute syntaktische Regeln eingeführt, um eine einheitliche und verständliche Darstellung in der konzeptionellen Phase mit Hilfe des ER-Modells sicherzustellen (Jarosch, 2016, S. 35):

1. Im zweigeteilten Kästchen eines Objekttyps werden die dem jeweiligen Objekttyp zugehörigen relevanten Eigenschaften im unteren Teil aufgelistet.
2. Die Reihenfolge der Eintragung der Attribute ist dabei bedeutungslos.
3. Die Benennung der relevanten Eigenschaft muss für den jeweiligen Objekttyp eindeutig sein und muss im Singular stehen.

Abbildung 4: Darstellung der Attribute im ER-Modell

In Abbildung 4 wird die grafische Darstellung innerhalb des ER-Modells gemäß den vorangegangen aufgezählten syntaktischen Regeln ersichtlich. Auf der linken Seite ist eine allgemeine Darstellung eines Objekttyps mitsamt seinen Attributen gegeben. Auf der rechten Seite ist ein konkretes Beispiel mit dem Objekttyp ‚Person‘ und seinen

relevanten Eigenschaften ‚Vorname', ‚Nachname', ‚Geburtsdatum' und ‚Geschlecht' zu sehen.

4.3 Identifizierung

Verschiedene Objekttypen untereinander sind bereits an ihrem von den Syntaxregeln aus Kapitel 4.1 geforderten eindeutigen Objekttypnamen unterscheidbar. Da sämtliche Objekte eines Objekttyps in gleicher Weise verarbeitet werden und, wie in Kapitel 4.2 beschrieben, dieselben Attribute vorweisen, müssen einzelne Objekte desselben Objekttyps unterscheidbar sein. Dies ist erforderlich, um gezielte Abfragen und die Verarbeitung der in den Attributen der Objekte gespeicherten Informationen zu ermöglichen.

Die Identifizierung eines Objekts innerhalb eines Objekttyps erfolgt in erster Linie über dessen Attributwerte. Dabei gibt es drei unterschiedliche Möglichkeiten, um Objekte eines Objekttyps anhand ihrer relevanten Eigenschaften zu identifizieren (Jarosch, 2016, S. 39). Bevor auf diese jedoch näher eingegangen wird, werden zunächst zwei verschiedene Arten von Eigenschaften voneinander abgegrenzt: Die identifizierende und die beschreibende Eigenschaft.

„Eine Eigenschaft ist eine *beschreibende Eigenschaft*, wenn sie zwar speicherwürdige Aspekte des jeweiligen Objekttyps beschreibt, jedoch nicht für die Identifizierung der Objekte des Objekttyps herangezogen wird." (Jarosch; 2016, S. 42)

Eine beschreibende Eigenschaft hält demnach nur deskriptive Informationen über das Objekt bereit, die sich nicht zur Identifizierung unter anderen Objekten desselben Objekttyps eignen. Im Gegensatz zu den deskriptiven Attributen stehen die identifizierenden Attribute (auch Schlüsselattribute genannt), welche identifizierende Eigenschaften besitzen:

„Eine Eigenschaft ist eine *identifizierende* (bzw. *teilidentifizierende*) *Eigenschaft*, wenn sie die Identifizierung eines Objekts innerhalb eines Objekttyps herbeiführt (bzw. zu dessen Identifizierung beiträgt)." (Jarosch, 2016, S. 41)

Die erste Möglichkeit, um Objekte eines Objekttyps anhand ihrer relevanten Eigenschaften zu identifizieren, ist die Identifizierung über eine unikale Eigenschaft. Hierbei ist eine Eigenschaft eines Objekttyps in einer Weise geartet, die garantiert, dass jeder mögliche annehmbare Eigenschaftswert dieser Eigenschaft nur bei einem einzigen Objekt des Objekttyps zur Ausprägung kommt. Durch diesen unikalen Eigenschaftswert existiert somit eine Bindung an lediglich ein einziges Objekt und

dieses kann anhand dieses Eigenschaftswertes eindeutig identifiziert werden (Jarosch, 2016, S. 39). Als veranschaulichendes Beispiel kann der Objekttyp ‚Auto' herangezogen werden. Als unikale Eigenschaft dieses Objekttyps ist das Kennzeichen zu nennen. Jedes einzelne Auto hat ein eindeutiges, sich von anderen unterscheidendes Kennzeichen, es ist an dieser einen unikalen Eigenschaft zweifelsfrei zu identifizieren.

Die zweite Möglichkeit involviert statt einer einzigen Eigenschaft gleich mehrere Eigenschaften desselben Objekttyps, um ein Objekt dieses Objekttyps identifizieren zu können. Dies wird getan, wenn eine Eigenschaft allein nicht in der Lage ist, eine eindeutige Identifizierung der Objekte vorzunehmen. Daher wird eine minimale Kombination aus (teil-)identifizierenden Eigenschaften als identifizierender Schlüssel verwendet. Dies ist eine Kombination von Eigenschaften, deren Eigenschaftswertekombination nur jeweils bei einem Objekt auftritt und somit unikal ist (Jarosch, 2016, S. 40). Als passendes Beispiel hierfür ist der Objekttyp ‚Person' zu nennen. Als einzige unikale Eigenschaft kommt der Nachname nicht in Betracht, da es mehrere Personen mit dem gleichen Nachnamen geben kann. Diesem Problem unterliegt auch das Attribut Vorname des Objekttyps Person. Eine Kombination der Eigenschaften Vorname und Nachname könnte immer noch mehrfach vergeben sein. Erst eine Kombination der Attribute Vorname, Nachname und Geburtsdatum lässt eine hinreichend eindeutige Identifizierung der Objekte des Objekttyps ‚Person' zu.

Es existiert jedoch noch eine dritte Möglichkeit, um eine Identifizierung von Objekten anhand der relevanten Eigenschaften eines Objekttyps vorzunehmen: Die Identifizierung unter Zuhilfenahme einer organisatorischen Eigenschaft. Oftmals ist es nicht möglich, eine natürliche unikale Eigenschaft oder eine natürliche Schlüsselkombination aus Attributen zu finden. Auch wenn dies zu umständlich wäre, beispielsweise zu viele Eigenschaften kombiniert werden müssen, um eine eindeutige Identifizierung der Objekte zu gewährleisten, wird die Variante der Identifizierung mit Hilfe einer organisatorischen Eigenschaft gewählt. Dazu wird eine neue künstliche Eigenschaft eingeführt, die den Zweck des identifizierenden Schlüssels übernimmt (Jarosch, 2016, S. 40; Geisler, 2014, S. 147). Als Beispiel einer solchen identifizierenden organisatorischen Eigenschaft kann bei dem Objekttyp ‚Rechnung' die künstliche Eigenschaft ‚Rechnungsnummer' verwendet werden. Dies kann beispielsweise eine fortlaufende Nummerierung der Rechnungen sein und ist somit für diesen Objekttyp unikal.

Der Unterschied der dritten Identifizierungsvariante zur erstgenannten besteht nur darin, dass statt einer natürlichen unikalen Eigenschaft auf eine künstlich angelegte

unikale Eigenschaft zurückgegriffen wird. Somit ergeben sich die zwei Hauptmethoden der Identifizierung von Objekten über eine einzige unikale Eigenschaft, sei diese natürlichen oder künstlichen Ursprungs, oder aber über eine Kombination aus Eigenschaften, die zusammen einen unikalen Schlüssel ergeben. Der Begriff *Schlüssel* ist, wie bereits zuvor verwendet, als identifizierende Eigenschaft oder Eigenschaftskombination zu verstehen.

Analog zu den Objekttypen und den Attributen gilt es auch bei der Identifizierung, erneut konsistente syntaktische Regeln einzuführen, die eine einheitliche und verständliche Darstellung im Rahmen des ER-Modells in der konzeptionellen Phase gewährleisten (Jarosch, 2016, S. 42):

1. Eine identifizierende bzw. mehrere teilidentifizierende Eigenschaften werden durch Unterstreichung kenntlich gemacht.
2. Die Position innerhalb der Liste der relevanten Eigenschaften im unteren Kästchen des Objekttyps einer (teil-)identifizierenden Eigenschaft ist irrelevant.

Die grafische Darstellung innerhalb des ER-Modells der identifizierenden Eigenschaften, auch Schlüssel genannt, ist in Abbildung 5 zu sehen. In der oberen Bildhälfte ist exemplarisch die Darstellung im Falle einer einzigen unikalen Eigenschaft dargestellt. Als Beispiel ist exemplarisch der Objekttyp ‚Auto' mit dessen Schlüssel ‚Kennzeichen' aufgeführt. Der Schlüssel ist durch die Unterstreichung erkennbar. In der unteren Bildhälfte hingegen ist neben dem allgemeinen Fall der Identifizierung durch eine Kombination von relevanten Eigenschaften exemplarisch der Objekttyp ‚Person' dargestellt, dessen Schlüssel aus der Attributkombination ‚Vorname', ‚Nachname' und ‚Geburtsdatum' besteht. Hierbei sind alle teilidentifizierenden Attribute, wie in den vorangegangenen Syntaxregeln gefordert, durch Unterstreichung des Attributnamens gekennzeichnet.

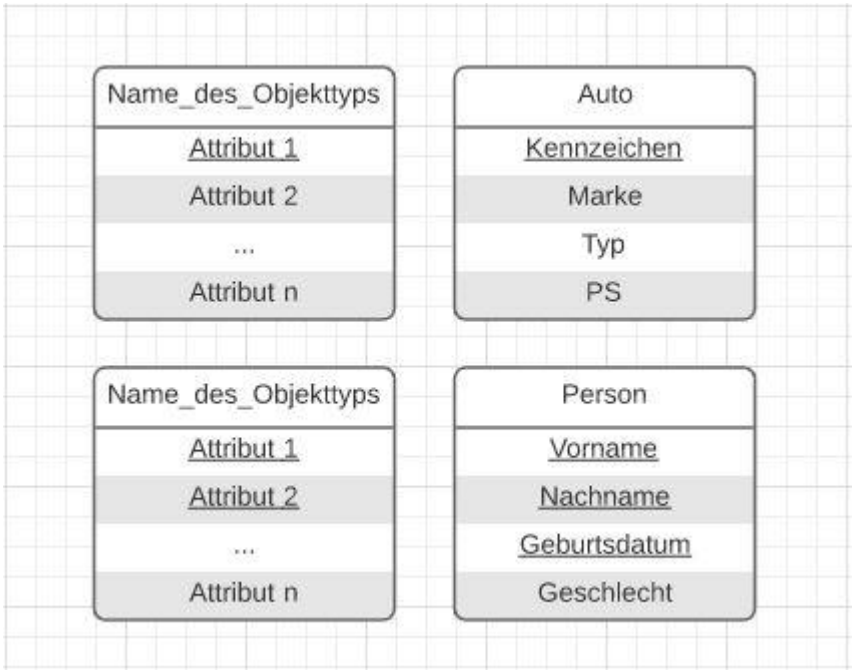


Abbildung 5: Darstellung der identifizierenden Eigenschaften im ER-Modell

4.4 Sachlogische Zusammenhänge

Bislang wurden innerhalb der Datenmodellierung mit Hilfe des ER-Modells lediglich zwei Konzepte beschrieben: die Objekttypen und Objekte als deren konkreten Realisierungen sowie deren relevante Eigenschaften (Attribute). Dies ist jedoch in den allermeisten Fällen nicht ausreichend, da in der Praxis Objekte in vielfältiger Weise miteinander in Verbindung stehen. Diese Zusammenhänge sind meist wesentlicher Bestandteil, ohne die das Datenmodell deutlich an Aussagekraft verliert und die vielfältigen Anforderungen nicht zielgemäß darstellbar sind (Jarosch, 2016, S. 45). Daraus folgt, dass auch diesen Zusammenhängen zwischen den Objekten Raum im konzeptionellen ER-Modell gegeben werden muss. Deswegen wird im Folgenden ein drittes Konzept des ER-Modells eingeführt: Die sachlogischen Zusammenhänge, auch *Beziehungen* oder auf Englisch *relationships* genannt. Das ER-Modell beschreibt damit einen Realitätsausschnitt als eine Menge von Objekten, übergeordnet gesammelt in Objekttypen, die durch Attribute näher definiert sind und durch Beziehungen untereinander verknüpft sind (Staud, 2005, S. 130).

Damit deckt sich auch die Definition einer Beziehung des Begründers des ER-Modells:

„A *relationship* is an association among entities.“ (Chen, 1976, S. 10)

Somit ist eine Beziehung eine Verbindung zwischen Entities, d.h. eine Verbindung zwischen Objekten bzw. Objekttypen. Dies ist insbesondere beim logischen Aufbau der

Daten und deren Abfrage und Verarbeitung relevant, da anhand dieser logischen Verknüpfungen Verbindungen erstellt werden, ohne die eine Abfrage und Verarbeitung der Daten oftmals nicht möglich oder nahezu unmöglich ist.

Konkrete Beziehungen zwischen konkreten ausgestalteten Objekten können analog zu den abstrakteren Objekttypen auch als Beziehungstypen auf abstrakterer Ebene zusammengefasst werden. Beziehungstypen verbinden Objekttypen anstatt konkrete Objekte (Schlageter und Stucky, 1983, S. 47; Saake et al., 2013, S. 61). Gleichartige Beziehungen zwischen konkreten Objekten der gleichen Objekttypen werden also zu Beziehungstypen zusammengefasst (Lang und Lockemann, 1995, S. 335). Im Rahmen des ER-Modells, das in dieser Studie für das konzeptionelle Design genutzt wird, sind hauptsächlich die allgemeineren Beziehungstypen von Relevanz und weniger die konkreten Ausprägungen der Beziehungen einzelner Objekte untereinander.

Des Weiteren sagt Chen (1976) über Beziehungen zwischen Objekten, dass die Rolle eines Objektes in einer Beziehung die Funktion ist, die es in dieser Beziehung erfüllt (Chen, 1976, S. 12). Unter Verwendung des bereits bekannten Beispiels der Objekttypen ‚Person' und ‚Auto' wäre ein Beziehungstyp zwischen ‚Person' und ‚Auto' denkbar, der die Besitzverhältnisse angibt: Eine Person kann ein Auto besitzen und ein Auto kann sich im Eigentum einer Person befinden. Der Person käme hierbei die Rolle des Eigentümers zu und dem Auto die Rolle des Eigentums.

Es ist nicht immer eindeutig, was Objekttyp oder was Beziehungstyp ist, dies wird ausschließlich von demjenigen determiniert, der die reale Welt in das konzeptionelle Datenmodell überführt (Schlageter und Stucky, 1983, S. 47). Sachverhalte der realen Welt können unterschiedlich aufgefasst und interpretiert werden, sodass beispielsweise einerseits der Objekttyp ‚Arbeitgeber' über eine Beziehung mit dem Objekttyp ‚Arbeitnehmer' verbunden sein kann. Andererseits ist auch die Ausgestaltung über einen dritten Objekttyp ‚Arbeitsverhältnis' denkbar, der wiederum über Beziehungstypen zu den Objekttypen ‚Arbeitnehmer' und ‚Arbeitgeber' verbunden ist.

In den folgenden beiden Kapiteln werden die sachlogischen Zusammenhänge zwischen den Objekten und Objekttypen in ihren speziellen Ausprägungen behandelt. Dazu wird eine Unterteilung in zwei Gruppen von Beziehungstypen vorgenommen: einerseits in duale Beziehungstypen und andererseits in Rekursiv-Beziehungstypen (Jarosch, 2016, S. 45). Im Anschluss daran werden der Vollständigkeit halber noch weitere Ausnahmen von Beziehungstypen in Kapitel 4.4.3 vorgestellt, die jedoch im Rahmen dieser Studie keine weitere Verwendung finden.

Anschließend werden die Optionalität und Kardinalität von Beziehungstypen (Kapitel 4.4.4) und ihre unterschiedlichen möglichen Ausprägungen näher erläutert. Zuletzt werden die syntaktischen Regeln zur Darstellung im ER-Modell dieser Studie dargelegt (Kapitel 4.4.5).

4.4.1 Duale Beziehungstypen

Wie bereits im vorhergehenden Abschnitt beschrieben wurde, ist eine Beziehung eine Verbindung bzw. ein sachlogischer Zusammenhang zwischen Objekten. Eine duale Beziehung „kennzeichnet den konkreten Zusammenhang zwischen zwei realen Objekten." (Jarosch, 2016, S. 46) Die zusätzliche Differenzierung liegt demnach in der Anzahl an beteiligten Objekten, nämlich genau zwei. In der Umgebung des konzeptionellen Datenmodells wird jedoch hauptsächlich mit Objekttypen und Beziehungstypen gearbeitet und weniger mit Objekten und konkreten Beziehungen zwischen diesen. Folglich ist eine Definition des dualen Beziehungstyps im Rahmen des ER-Modells dieser Studie unumgänglich: „Ein (*dualer*) *Beziehungstyp* (engl. relationship type) beschreibt den typmäßigen sachlogischen Zusammenhang, der zwischen den Objekten zweier Objekttypen besteht." (Jarosch, 2016, S. 47)

Diese dualen Beziehungstypen machen den Hauptteil an Beziehungstypen im konzeptionellen Datenmodell dieser Studie aus. Aus diesem Grund wird im Folgenden ein dualer Beziehungstyp vereinfachend als *Beziehungstyp* bezeichnet. Sollte es sich um einen anderen Beziehungstyp handeln, wird dieser durch ein geeignetes vorangestelltes Wort kenntlich gemacht.

4.4.2 Rekursiv-Beziehungstypen

Mitunter kann es vorkommen, dass eine sachlogische Verbindung zwischen zwei Objekten desselben Objekttyps modelliert werden muss. Beispielsweise eine Verbindung zwischen Person A und Person B, die beide demselben Objekttyp ‚Person' angehören. Dies ist mit der Einführung einer neuen Beziehungstyp-Variante möglich (Schlageter und Stucky, 1983, S. 49). Sachlogische Zusammenhänge dieser Art werden *rekursive Beziehungstypen* genannt und können wie folgt definiert werden: „Ein *Rekursiv-Beziehungstyp* (engl. recursive relationship type) beschreibt den sachlogischen Zusammenhang zwischen Objekten, die demselben Objekttyp angehören." (Jarosch, 2016, S. 65)

Im Fall eines Rekursiv-Beziehungstyps reicht es jedoch nicht aus, wie beim dualen Beziehungstyp den Beziehungstyp-Richtungen lediglich die Benennung der teilhabenden Objekttypen anzugeben, da beide Verbindungen den gleichen Namen

(den des einzigen beteiligten Objekttyps) tragen würden. Stattdessen sind die beiden Verbindungen des Beziehungstyps zum Objekttyp mit den erwähnten Rollen, welche die Objekte einnehmen, zu betiteln (Lang und Lockemann, 1995, S. 336). Ausgehend vom bereits bekannten Beispiel des Objekttyps ‚Person', wäre ein Rekursiv-Beziehungstyp denkbar, der die Verwandtschaftsverhältnisse zweier konkreter Personen angibt. Sei Person A ein Elternteil von Person B. So würden beispielsweise die Rollennamen ‚Elternteil' oder ‚ist Elternteil von' bzw. ‚Kind' oder ‚ist Kind von' an den entsprechenden Verbindungen zwischen Beziehungstyp und Objekttyp angebracht werden.

Zur leichteren Verständlichkeit sind im Rahmen dieser Studie auch einige duale Beziehungstypen anstatt mit den verbundenen Objekttypen mit den Rollennamen gekennzeichnet, die diese in dem entsprechenden Beziehungstyp einnehmen.

4.4.3 AUSNAHMEN

Neben der genannten Definition der dualen Beziehung aus Kapitel 4.4.1 finden sich weitere Definitionen einer Beziehung im ER-Modell, die deutlich weiter und umfassender gefasst sind. Als Beispiel und Vertreter dieser ist hierfür die Definition von Lang und Lockemann (1995) zu nennen:

„Eine *Beziehung* (Relationship) beschreibt einen Zusammenhang zwischen mehreren Gegenständen und reichert diesen gegebenenfalls durch Information an. Beziehungen gleicher Art, d.h. zwischen Gegenständen der gleichen Typen und mit gleichen Attributen, werden zu *Beziehungstypen* (engl. relationship types) zusammengefaßt." (Lang und Lockemann, 1995, S. 335)

Auch andere Autoren schließen sich den Parametern dieser Definition an (Schlageter und Stucky, 1983, S. 48; Geisler, 2014, S. 156ff.). In diesen Fällen geht die Definition über die Beteiligung von nur zwei Objekttypen hinaus und erlaubt auch eine Beteiligung von drei oder mehr Objekttypen an einem Beziehungstyp. Des Weiteren ist in dieser Definition mit eingeschlossen, dass ein Beziehungstyp, analog zu einem Objekttyp, über Attribute verfügen kann.

Da Beziehungen mit mehr als zwei Objekten im Anwendungsfall dieser Studie nicht verwendet werden und die Einbindung von dem Beziehungstyp zugehörigen Attributen in einigen ER-Dialekten sogar untersagt ist (Lang und Lockemann, 1995, S. 335), wird nicht näher auf diese eingegangen.

4.4.4 Optionalität und Kardinalität von Beziehungstypen

Um Beziehungstypen aussagekräftig zu machen, bedarf es noch der Definition und Erklärung der Begriffe *Optionalität* und *Kardinalität* einer Beziehungstyp-Richtung. Unter der Verwendung des Beispiels des Eigentums einer Person an einem Auto stellen sich beispielsweise die Fragen, ob eine Person zwangsweise ein Auto besitzen muss oder auch keines besitzen kann. Des Weiteren ist bisher nicht definiert, ob eine Person mehr als nur ein Auto besitzen kann. In ersterem Fall geht es um die *Optionalität*, in letzterem um die *Kardinalität* der Beziehungstyp-Richtung.

Zur umfassenden Beschreibung komplexer Beziehungstypen zwischen Objekttypen ist für jede Verbindungsrichtung die Angabe der *Kardinalität* erforderlich. Sie gibt Aussage darüber, welche Höchst- und Mindestanzahl an Beziehungsinstanzen ein Objekt eines Objekttyps eingehen kann (Lang und Lockemann, 1995, S. 337; Elmasri und Navathe, 2011, S. 73). Grundsätzlich gibt es die drei folgenden Kardinalitätsverhältnisse bei Beziehungstypen: 1:1-, 1:N- und N:M-Beziehungen (Staud, 2005, S. 131). Die Bezeichnungen geben die Verhältnisse der Anzahl an Beziehungsinstanzen von Objekten an. ‚1' steht dafür, dass genau eine Beziehung von diesem Objekt eingegangen wird. ‚N' steht für eine beliebige Anzahl, auch größer eins (Jarosch, 2016, S. 48). Die Bezeichnung ‚M' ist analog zur Bezeichnung ‚N' zu verstehen und ist nur der Übersichtlichkeit halber mit einem anderen Buchstaben versehen. Als passendes Beispiel zur verständlicheren Anschauung können wieder die Objekttypen ‚Auto' und ‚Person' verwendet werden. Eine 1:1-Beziehung zwischen Objekten dieser Objekttypen bedeutet in diesem Fall, dass genau eine Person mit einem Auto eine Beziehung eingeht, es beispielsweise besitzt. Kann eine Person mehrere Autos besitzen, handelt es sich um eine 1:N-Beziehung. Dies wäre auch umgekehrt denkbar, indem ein Auto im Besitz mehrerer Personen ist. Sind beide Fälle möglich, befindet sich beispielsweise ein Auto im Besitz mehrerer Personen und eine Person kann mehrere Autos besitzen, so ist im ER-Modell ein N:M-Beziehungstyp zwischen den beiden Objekttypen zu wählen.

Bei der Optionalität geht es um die Frage, ob ein Objekt eines Objekttyps optional oder nicht-optional an einem Beziehungstyp beteiligt ist, also ob ein Objekt mindestens eine Beziehung eingehen muss (nicht-optional) oder kann (optional) (Jarosch, 2016, S. 48; Elmasri und Navathe, 2011, S. 74). Im Rahmen des ER-Modells wird die Optionalität einer Beziehungstyprichtung mit einem vorangestellten ‚C' gekennzeichnet. Unter erneuter Verwendung des Beispiels der Beziehung zwischen Objekten der Objekttypen ‚Auto' und ‚Person' zur besseren Veranschaulichung würde eine optionale Beziehung auf Seiten der Person bedeuten, dass diese ein (mehrere) Auto(s) besitzen kann oder

aber auch keines. Wenn von Seiten des Autos die Beziehung nicht-optional gegeben ist, dann gehört ein Auto immer einer Person. Es handelt sich in diesem Beispiel also um eine 1:C1-Beziehung (1:CN-Beziehung) zwischen den Objekttypen ‚Person' und ‚Auto'.

Aus der Kombination von Optionalität und Kardinalität ergeben sich 16 mögliche Kardinalitätskombinationen von Beziehungstypen im ER-Modell, die im Klassifikationsschema in Tabelle 1 aufgeführt sind. Hierbei sind jedoch auch die Spiegelbilder, wie zum Beispiel 1:N und N:1, mitgezählt, sodass sich abzüglich dieser lediglich 10 symmetriefreie Klassen ergeben. Diese befinden sich im oberen Rechten, nichtausgegrauten Bereich des Klassifikationsschemas.

Tabelle 1: Kardinalität von Beziehungstypen (In Anlehnung an Jarosch, 2016, S. 51)

Beziehungstyp-Richtung	A => B	Kardinalität 1	Kardinalität 1	Kardinalität N	Kardinalität N
B => A		Nicht optional	Optional	Nicht-optional	optional
Kardinalität 1	Nicht optional	1:1	1:C1	1:N	1:CN
Kardinalität 1	optional	C1:1	C1:C1	C1:N	C1:CN
Kardinalität N	Nicht-optional	N:1	N:C1	M:N	M:CN
Kardinalität N	optional	CN:1	CN:C1	CM:N	CM:CN

Eine Anwendung dieser 10 symmetriefreien Klassen von Beziehungstypen lässt sich jedoch nicht ohne weiteres auf rekursive Beziehungstypen übertragen. Denn rekursive Beziehungstypen liegen, wie in Kapitel 4.4.2 beschrieben wurde, zwischen Objekten desselben Objekttyps vor. Da der übergeordnete Objekttyp identisch ist, liegen auch keine zwei Objekttypen mit unterschiedlicher Anzahl an Objekten vor, sodass die Klassifizierung hinsichtlich Kardinalität und Optionalität des Rekursiv-Beziehungstyps geprüft und angepasst werden muss. In Tabelle 2 werden die 10 symmetriefreien Klassen des dualen Beziehungstyps mit den daraus resultierenden Bedingungen an das Größenverhältnis zwischen der Anzahl der beiden am dualen Beziehungstyp beteiligten Objekte von den exemplarischen und abstrakten Objekttypen A und B dargestellt sowie die Möglichkeit der Anwendung der Klasse auf den rekursiven Beziehungstyp abgeleitet.

Tabelle 2: Mögliche Klassen von Rekursiv-Beziehungstypen (In Anlehnung an Jarosch, 2016, S. 69)

Beziehungstyp-Klasse	Bedingung für das Größenverhältnis	Rekursiv-Beziehungstyp
1:1	|A| = |B|	möglich
1:C1	|A| > |B|	nicht möglich
C1:C1	keine	möglich
1:N	|A| < |B|	nicht möglich
C1:N	|A| < |B|	nicht möglich
1:CN	keine	möglich
C1:CN	keine	möglich
M:N	keine	möglich
M:CN	keine	möglich
CM:CN	keine	möglich

Besteht ein 1:1-Beziehungstyp zwischen den exemplarischen Objekttypen A und B, dann muss jedes Objekt des Objekttyps A eine Beziehung mit einem Objekt des Objekttyps B eingehen - und umgekehrt. Daraus folgt, dass die Anzahl an Objekten des Objekttyps A der Anzahl an Objekten des Objekttyps B entsprechen muss. Da im Fall eines Rekursiv-Beziehungstyps A und B identisch wären, ist das geforderte Mengenverhältnis gegeben und somit ist diese Klasse für den Rekursiv-Beziehungstyp möglich.

Gegeben den Fall einer 1:C1-Beziehungstyp-Klasse, wie es in Zeile zwei der Tabelle 2 ersichtlich ist, muss jedes Objekt des Objekttyps B eine Beziehung mit einem Objekt vom Objekttyp A eingehen, wohingegen ein Objekt des Objekttyps A auch keine bzw. mehrere Beziehungen mit Objekten des Objekttyps B eingehen kann. Dies ist nur möglich, wenn die Anzahl der Objekte vom Objekttyp A größer ist als die der Objekte vom Objekttyp B. Diese Klasse ist folglich beim Rekursiv-Beziehungstyp nicht möglich. Abschließend lässt sich festhalten, dass immer wenn eine Beziehungstyp-Klasse ein ungleiches Mengenverhältnis fordert, diese nicht auf den Rekursiv-Beziehungstyp anwendbar ist.

4.4.5 Darstellung von Beziehungstypen im ER-Modell

Auch für die Darstellung der Beziehungstypen im konzeptionellen ER-Modell mitsamt den Angaben bezüglich Optionalität und Kardinalität bedarf es einer einheitlichen Verwendung im Rahmen dieser Studie, um eine durchgängig verständliche und übersichtliche Nachvollziehbarkeit zu gewährleisten. Entsprechende syntaktische

Regeln wurden bereits von Jarosch (2016) aufgestellt und erfahren nun eine gewisse Anpassung, um ihre Verwendung mit den im Zuge dieser Studie genutzten Programmen zur Darstellung des ER-Modells zu ermöglichen (Jarosch, 2016, S. 49):

1. Ein Beziehungstyp zwischen zwei Objekttypen wird als Verbindungslinie zwischen diesen dargestellt.
2. Die Beziehungstyp-Richtung vom ersten Objekttyp zum zweiten Objekttyp steht auf der Seite des ersten Objekttyps. Dies gilt umgekehrt für die andere Beziehungstyp-Richtung.
3. Die Optionalität einer Beziehungstyp-Richtung wird durch einen leeren Kreis am Ende der Verbindungslinie dargestellt. Ist sie mit keinem Kreis versehen, dann ist diese Beziehungstyp-Richtung nicht-optional und damit obligatorisch.
4. Beträgt die Kardinalität einer Beziehungstyp-Richtung den Wert 1, dann wird dies durch einen einfachen Querstrich am Ende der Verbindungslinie gekennzeichnet. Die Kardinalität N hingegen wird durch drei auseinanderstrebende Striche am Ende der Verbindungslinie repräsentiert.
5. Existieren minimale oder maximale Grenzen der Kardinalität N, dann werden diese innerhalb eckiger Klammern vor der Benennung der Beziehungstyp-Richtung grafisch festgehalten.

In Abbildung 6 sind exemplarisch zwei Beziehungstypen zwischen Objekttypen mitsamt Angaben zu Optionalität und Kardinalität der Beziehungstyp-Richtungen abgebildet. Auf der linken Seite ist ein 1:N-Beziehungstyp dargestellt, welcher durch die Verbindungslinie zwischen den Objekttypen A und B kenntlich gemacht ist. Hierbei symbolisiert der Querstrich am Ende der Verbindung auf der Seite des Objekttyps A die Kardinalität 1 und das in drei Linien aufgegliederte Ende der Verbindung auf Seiten des Objekttyps B die Kardinalität N. Des Weiteren sind in der linken Abbildungshälfte in eckigen Klammern die minimale und die maximale Anzahl an sachlogischen Zusammenhängen des Objekttyps A mit dem Objekttyp B dargestellt. Sollten solche Bedingungen existieren, dann werden diese in eckigen Klammern aufgeführt. In der rechten Bildhälfte ist analog ein Beispiel zu sehen, bei dem der Beziehungstyp in beiden Richtungen mit einem Kreis an den Enden der Verbindungslinie gekennzeichnet und somit als optional ausgewiesen ist. In diesem Beispielfall handelt es sich um einen C1:CN-Beziehungstyp zwischen den Objekttypen C und D.

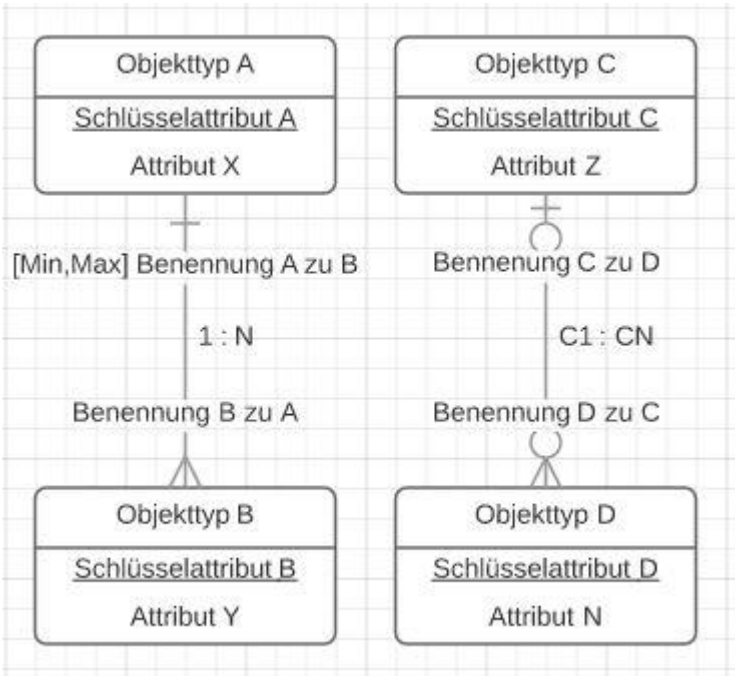


Abbildung 6: Darstellung von Beziehungstypen im ER-Modell

Im Rahmen dieser Studie wird eine Notation des ER-Modells verwendet, die auch unter dem Namen *Crow's-Foot-Darstellung* bekannt ist. Dieser Name weist auf die drei sich in Linien aufspaltenden Enden der Verbindungslinie hin, wenn mehrere Beziehungen möglich sind. Da dies einem Krähenfuß ähnelt, erhält diese Darstellung ihren Namen (Geisler, 2014, S. 150).

4.5 Qualitätssicherung konzeptioneller Datenmodelle

Die vielfältigen Elemente des ER-Modells und deren grafische Darstellung wurden in den vorhergehenden Abschnitten eingehend erläutert. Unter Einhaltung dieser Konventionen ist jedoch noch keine Aussage über die Qualität des Datenmodells zu treffen. Die objektive Qualitätsüberprüfung eines konzeptionellen Datenmodells bedient sich dem Prozess der Normalisierung. Die Normalformen wurden zwar für das relationale Datenbankmodell entworfen, sind aber auch auf der konzeptionellen Ebene bereits sinnvoll anwendbar (Jarosch, 2016, S. 91). Im Rahmen dieser Studie werden im Kontext des konzeptionellen Datenmodells die ersten drei Normalformen als Kriterien zur Qualitätsüberprüfung herangezogen. Die einzelnen Normalformen sind aufeinander aufbauend. Ist also die n-te Normalform erfüllt, so trifft dies auch auf alle „niedrigeren" Normalformen zu (Staud, 2005, S. 48). Im Folgenden werden die ersten drei Normalformen vorgestellt.

4.5.1 Erste Normalform

Die erste Normalform hat zum Ziel, dass kein Objekttyp des Datenmodells mehr als eine multiple Eigenschaft (Attribut) besitzt. Es kommt also erst zur Erfüllung der ersten Normalform, wenn das Datenmodell keine multiplen Eigenschaften mehr vorweist (Jarosch, 2016, S. 93ff.). In einem veranschaulichenden Beispiel mit dem Objekttyp ‚Auto' könnte die Eigenschaft ‚Farbe' demnach nicht gleichzeitig mit ‚Rot' und ‚Blau' belegt sein. Unter der Verwendung des Beispiels mit dem Objekttyp ‚Person' und seinem Attribut ‚Adresse' kann eine Person keine mehreren Adressen haben, um in der ersten Normalform vorzuliegen. Um es dennoch möglich zu machen, dass eine Person mehrere Adressen haben kann, beispielsweise Erst- und Zweitwohnsitz, muss das Attribut ‚Adresse' des Objekttyps ‚Person' zu einem eigenen Objekttyp ausgegliedert werden, der mit dem Objekttyp ‚Person' über einen 1:N-Beziehungstyp verknüpft ist.

4.5.2 Zweite Normalform

Die zweite Normalform ist erreicht, wenn „jedes Nichtschlüsselattribut voll funktional abhängig ist vom (gesamten) Schlüssel." (Staud, 2005, S. 70) Eine Eigenschaft B ist von Eigenschaft A desselben Objekttyps dann „funktional abhängig, wenn sich für jedes konkrete Objekt dieses Objekttyps aus dem Wert der Eigenschaft A direkt auf den Wert der Eigenschaft B schließen lässt." (Jarosch, 2016, S. 97)

Eine Verletzung der zweiten Normalform kann folglich also nur vorkommen, wenn ein Objekttyp einen zusammengesetzten Schlüssel besitzt, denn nur in dieser Konstellation kann es vorkommen, dass eine beschreibende Eigenschaft bereits von einem Teilschlüssel statt vom Gesamtschlüssel funktional abhängig ist (Jarosch, 2016, S. 99). Unter anderem aus diesem Grund wird im Rahmen dieser Studie auf die Verwendung von zusammengesetzten Schlüsseln verzichtet und – wo es nötig ist – eine künstliche organisatorische Eigenschaft, wie in Kapitel 4.3 beschrieben wurde, als Schlüssel eingeführt.

4.5.3 Dritte Normalform

Die dritte Normalform fordert, dass „keine beschreibende Eigenschaft eines Objekttyps vom Schlüssel dieses Objekttyps transitiv funktional abhängig ist." (Jarosch, 2016, S. 104) Transitive funktionale Abhängigkeit bedeutet, dass eine Eigenschaft innerhalb eines Objekttyps nicht direkt, sondern über eine weitere Eigenschaft funktional abhängig ist (Jarosch, 2016, S. 103f.). Ein Attribut C eines Objekttyps ist also dann transitiv funktional von Attribut A desselben Objekttyps abhängig, wenn Attribut C von Attribut B desselben Objekttyps funktional abhängig ist, und Attribut B wiederum

funktional abhängig von Attribut A ist. Vereinfacht ausgedrückt besteht transitive funktionale Abhängigkeit zwischen C und A, wenn sowohl C von B funktional abhängig ist, als auch B von A funktional abhängig ist.

4.6 Konzeptionelles Modell im Anwendungsfall

In diesem Abschnitt wird ein konzeptionelles ER-Modell, im Einklang mit den in den Kapiteln 4.1 bis 4.5 beschriebenen Grundsätze und Regeln für ein ER-Modell, für eine Erfahrungsdatenbank des Expertensystems für den Rückbau kerntechnischer Anlagen erstellt. Dies geschieht auf Basis der Anforderungen an die Erfahrungsdatenbank, die aus der durchgeführten Anforderungsanalyse in Kapitel 3 abgeleitet wurden.

4.6.1 Objekttypen und deren Attribute

Zuerst findet die Betrachtung der im konzeptionellen Modell zu erstellenden Objekttypen inklusive ihrer speicherwürdigen Informationen (ihren Attributen) statt. Als wichtiges Element wurde in der Anforderungsanalyse aus Kapitel 3 der Reaktortyp identifiziert, da dieser je nach Ausgestaltung mit einem Unterschied in den auszuführenden Rückbauvorgängen einhergeht. Analog zu den im Rahmen dieser Studie in Kapitel 4 etablierten Regeln und Konventionen des ER-Modells ist in Abbildung 7 der Reaktortyp als Objekttyp mitsamt seinen relevanten Attributen dargestellt. Als identifizierendes Element wurde sich in diesem Fall einer künstlichen organisatorischen Eigenschaft bedient, die an der Unterstreichung erkennbar ist. Die künstliche Einheit als Schlüssel wurde gewählt, um einerseits die Normalformen einzuhalten und andererseits um einen einfachen und nicht einen zusammengesetzten Schlüssel zu erhalten. Neben dem zum Objekttyp gleichnamigen Attribut ‚Reaktortyp' ist laut Anforderungsanalyse auch dessen Generation festzuhalten, da auch diese von Bedeutung für zu ergreifenden Rückbaumaßnahmen ist.

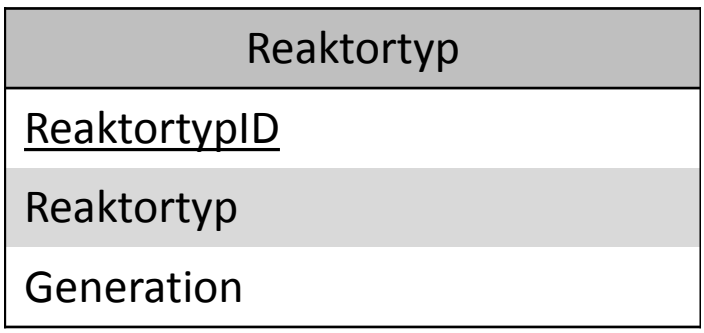


Abbildung 7: Objekttyp ‚Reaktortyp' mit Attributen im konzeptionellen ER-Modell

Neben den Details zum rückgebauten Reaktortyp sind in der Erfahrungsdatenbank auch allgemeine Daten zum Rückbauprojekt zu speichern. Neben dem identifizierenden Projektnamen sind hierbei vor allen Dingen die Restbetriebskosten zu Projektbeginn und das Fälligkeitsdatum relevant. Diese Eigenschaften werden durch die Attribute ‚Restbetriebskosten' und ‚DueDate' (engl. für Fälligkeitsdatum) im Objekttyp ‚Projekt' abgebildet (vgl. Abbildung 8).

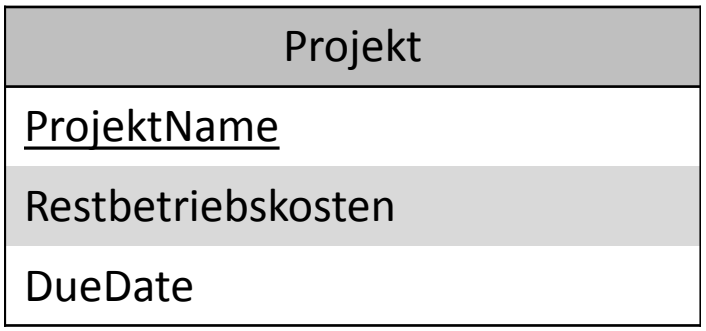


Abbildung 8: Objekttyp ‚Projekt' mit Attributen im konzeptionellen ER-Modell

Des Weiteren hat die Anforderungsanalyse aus Kapitel 3 ergeben, dass eine Berücksichtigung der vorhandenen und bei der Durchführung des Rückbauprojekts in Anspruch genommenen Ressourcen in der Erfahrungsdatenbank notwendig ist. Deshalb wird im konzeptionellen ER-Modell der in Abbildung 9 dargestellte Objekttyp ‚Ressource' eingeführt. Auch hier ist erneut eine organisatorische Eigenschaft als identifizierendes Attribut gewählt worden, das durch Unterstreichung kenntlich gemacht ist. Des Weiteren sind die Bezeichnung der Ressource und Daten bzgl. der Investitionskosten bei der Anschaffung sowie die variablen Kosten wichtige zu speichernde Informationen. Darüber hinaus wird die Art der Ressource (wie in Kapitel 3 beschrieben wurde) im Attribut ‚Typ' gespeichert. Die maximale Anzahl verfügbarer Ressourcen für ein Projekt wird durch das Attribut ‚MaxVerfügbar' angegeben. Bei den meisten Ressourcen kann hier ein beliebig großer Wert angegeben werden, wenn eine beliebige Anzahl an Ressourcen beschafft werden kann. In einigen Fällen ist die Verfügbarkeit allerdings begrenzt, wie beispielsweise bei der Ressource ‚Reaktorkran' oder bei bestimmten Behältnissen.

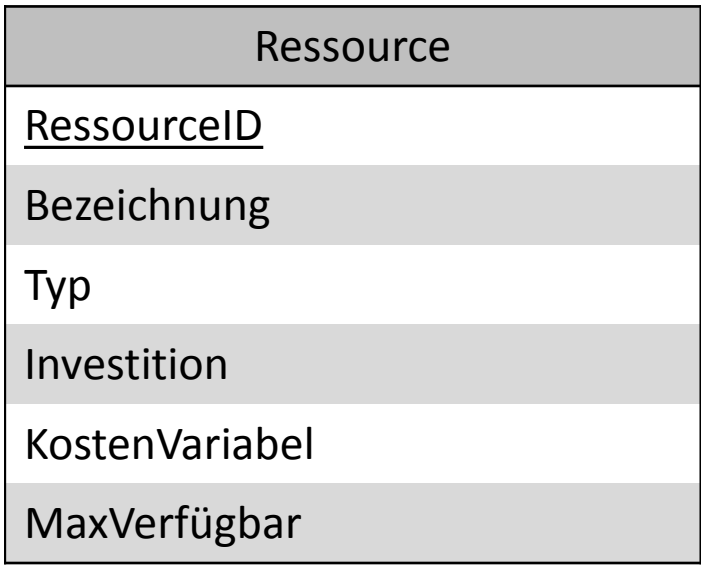


Abbildung 9: Objekttyp ‚Ressource' mit Attributen im konzeptionellen ER-Modell

Den Kern der Erfahrungsdatenbank stellen die Vorgänge dar, die im Rahmen eines Rückbauprojekts durchzuführen sind. Die Anforderungsanalyse aus Kapitel 3 hat ergeben, dass für Vorgänge neben organisatorischen Eigenschaften, wie dem künstlichen identifizierenden Attribut ‚VorgangID', und deskriptiven Eigenschaften, wie der Bezeichnung eines Vorgangs, auch weitere Informationen festgehalten werden müssen. Diese sind neben dem Ausführungsort und der Ausführungswahrscheinlichkeit eines Vorgangs auch die Strahlungsintensität, die bei der Durchführung vorherrscht, sowie die Verminderung der Restbetriebskosten, die durch die Ausführung des Vorgangs resultiert. In Abbildung 10 sind neben den eben genannten Attributen außerdem weitere Attribute aufgeführt, die den Objekttypen ‚Vorgang' und ‚Modus' angehören. Beim Objekttyp ‚Vorgang' muss laut Anforderungsanalyse zusätzlich die Kennzeichnung von Vorgängen möglich sein, die wiederholt in Zyklen durchgeführt werden können. Falls es eine maximale Anzahl an Zyklen gibt, muss diese speicherbar und auslesbar sein. Im Attribut ‚MaxZyklus' kann daher die maximale Anzahl an durchführbaren Zyklen festgehalten werden. Wird das Attribut mit dem Wert 1 belegt, dann kann daraus geschlossen werden, dass der Vorgang nicht wiederholt angewendet werden kann. Ist der belegte Wert größer als 1, bedeutet dies, dass der Vorgang so oft zyklisch wiederholt werden kann, bis die maximale Anzahl an Durchläufen erreicht wurde. Da ein Vorgang mehrere Ausführungsvarianten haben kann, wird neben dem Vorgang noch ein weiterer, ebenso in Abbildung 10 gezeigter, Objekttyp im konzeptionellen Modell der Erfahrungsdatenbank eingeführt: der Objekttyp ‚Modus'. In ihm sind neben dem organisatorischen Identifikationsattribut ‚ModusID' die Modusnummer, die Dauer und die Information, ob es sich bei der Dauer um einen Ist- oder einen Planwert handelt,

hinterlegt. Die Modusnummer ist ein Indikator dafür, um welchen Modus eines Vorgangs es sich bei konkreter Ausgestaltung der Objekttypen handelt.

Vorgang
VorgangID
Bezeichnung
Ausführungsort
Wahrscheinlichkeit
VerminderungRestbetriebskosten
MaxZyklus
Strahlung
Bemerkung

Modus
ModusID
ModusNummer
IstPlan
Dauer

Abbildung 10: Objekttypen ‚Vorgang' und ‚Modus' mit Attributen im konzeptionellen ER-Modell

Des Weiteren ist festzuhalten, welche Ressourcen in welcher Höhe von einem Vorgang in einem bestimmten ausgeführten Modus beansprucht werden. Dazu ist ein weiterer Objekttyp einzuführen. Der Objekttyp ‚Allokation' hält durch sein Attribut ‚Menge' die beanspruchte Höhe der jeweiligen Ressource fest. Wie in Abbildung 11 zu sehen ist, fungiert auch hier wieder eine künstliche organisatorische Eigenschaft als identifizierendes Element.

Allokation
AllokationID
Menge

Abbildung 11: Objekttyp ‚Allokation' mit Attributen im konzeptionellen ER-Modell

Von projektplanerischer Bedeutung sind darüber hinaus die zeitlichen Abhängigkeitsverhältnisse zwischen den Vorgängen. Diese werden in einem weiteren Objekttyp innerhalb der zu entwickelnden Datenbank angelegt. Abbildung 12 zeigt den Objekttyp ‚Anordnung', der durch sein Attribut ‚Art' die Art der Anordnungsbeziehung zwischen Vorgänger- und Nachfolgervorgängen festhält. Des Weiteren ist dem Objekttyp das organisatorische Schlüsselattribut ‚AnordnungID' hinzugefügt worden.

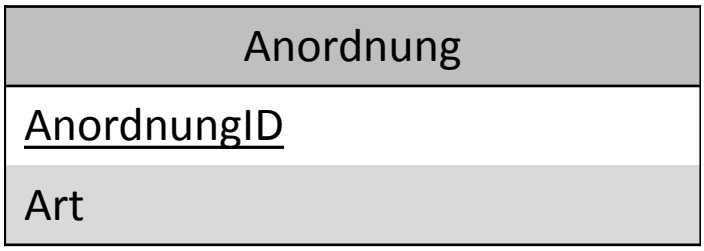


Abbildung 12: Objekttyp 'Anordnung' mit Attributen im konzeptionellen ER-Modell

4.6.2 SACHLOGISCHE ZUSAMMENHÄNGE

Nicht nur die Objekttypen in Verbindung mit ihren jeweiligen Attributen sind Bestandteil des konzeptionellen Modells, sondern auch sind die Verknüpfungen der Objekttypen untereinander. Ohne diese Verknüpfungen fehlen viele Informationen, die allein aus den im vorigen Abschnitt eingeführten Objekttypen nicht ersichtlich sind. Diese sachlogischen Zusammenhänge innerhalb des konzeptionellen ER-Modells wurden in Kapitel 4.4 näher erläutert.

Aus der Anforderungsanalyse in Kapitel 3 ist die Notwendigkeit einer Vielzahl solcher Beziehungen zwischen Objekttypen sichtbar geworden. Beispielsweise müssen die Anordnungsbeziehungen eingeführt werden, die den Objekttyp ‚Anordnung' mit den Vorgängen aus dem Objekttyp ‚Vorgang' in Verbindung bringen, um sicher wissen zu können, welcher Vorgang welche Nachfolger bzw. Vorgänger hat. Dabei kann ein Vorgang keinen oder mehrere Nachfolger bzw. Vorgänger haben, denen er zeitlich hinten- bzw. vorne angestellt ist. Die Anordnungsbeziehung, die mit ihrer Art im Objekttyp ‚Anordnung' gespeichert ist, muss jedoch immer mit einem Vorgang, der als Vorgänger und einem der als Nachfolger fungiert, in Verbindung stehen. In diesem Fall handelt es sich also um zwei 1:CN–Beziehungstypen, die jeweils vom Objekttyp ‚Vorgang' zum Objekttyp ‚Anordnung' gehen. Dabei verbindet einer dieser Beziehungstypen die Vorgänger und der andere die Nachfolger. Diese beiden und sämtliche im Folgenden beschriebenen Beziehungstypen sind in Abbildung 13 im Kontext des gesamten konzeptionellen Datenschemas zu sehen.

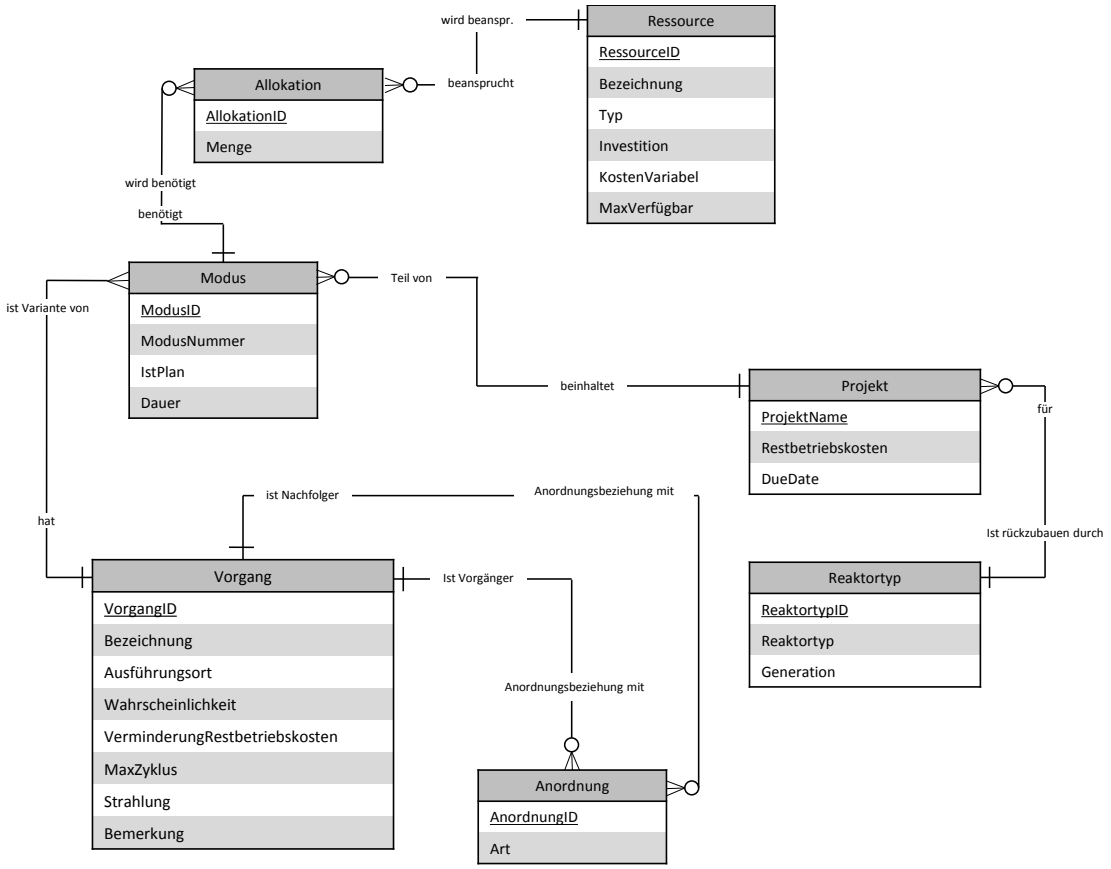


Abbildung 13: Konzeptionelles Modell der Erfahrungsdatenbank für den Rückbau kerntechnischer Anlagen

Die Beziehungen mit dem Objekttyp ‚Anordnung' sind nicht die einzigen Beziehungstypen, die der Objekttyp ‚Vorgang' eingeht. Auch mit dem Objekttyp ‚Modus' besteht eine Verbindung, da dieser mögliche Ausführungsvarianten eines Vorgangs beschreibt. Folglich handelt es sich hierbei um einen 1:N-Beziehungstyp, da ein Vorgang mindestens eine Ausführungsvariante haben muss, aber auch in mehreren Varianten ausführbar sein kann. Dahingegen muss ein Modus immer Variante eines Vorgangs sein.

Auch eine Verbindung zwischen den Objekttypen ‚Projekt' und ‚Modus' muss gegeben sein, um die Ausführungsvarianten eines Vorgangs sowie der tatsächlich ausgeführte Modus einem Rückbauprojekt zuordnen zu können. Im konzeptionellen ER-Modell wird diese Verbindung durch einen 1:CN-Beziehungstyp in Richtung Projekt zu Modus realisiert. Jeder geplante oder tatsächlich ausgeführte Modus muss eindeutig einem Projekt zugeordnet werden. Umgekehrt muss ein Projekt mit mehreren Modi in Verbindung stehen können, da ein Projekt in der Regel mehrere Vorgänge und deren Ausführungsvarianten umfasst. Es soll aber auch möglich sein, ein neues Projekt anzulegen, ohne diesem im selben Schritt bereits Modi zuordnen zu müssen, weshalb der Beziehungstyp in dieser Richtung optional ist.

Ausgehend vom Objekttyp ‚Projekt' ist eine Verbindung zum Projekttyp ‚Reaktortyp' relevant. Dies wird durch einen weiteren 1:CN-Beziehungstyp modelliert. Die Beziehungstyp-Richtung geht dabei jedoch vom Objekttyp ‚Reaktortyp' zum Objekttyp ‚Projekt'. Ein Rückbauprojekt muss immer einem rückzubauenden Reaktortyp zugeordnet werden, da der Reaktortyp signifikante Auswirkungen an das Herangehen an den Rückbau hat. Andererseits kann es mehrere Rückbauprojekte kerntechnischer Anlagen geben, die mit demselben Reaktortyp ausgestattet sind. Aber auch hier soll es wiederum möglich sein, einen Reaktortyp abzuspeichern, ohne ihm direkt ein Projekt zuordnen zu müssen. Aus diesem Grund besteht eine Optionalität der einen Beziehungstyp-Richtung.

Die Ressourcen und deren Allokation müssen mit den anderen Objekttypen in Beziehung gesetzt werden. Innerhalb der Erfahrungsdatenbank soll abgebildet werden können, welcher Modus innerhalb eines Projekts welche Ressourcen in welcher Höhe beansprucht. Dazu sind zum einen ein 1:CN-Beziehungstyp vom Objekttyp ‚Modus' zum Objekttyp ‚Allokation' und zum anderen ein weiterer 1:CN-Beziehungstyp vom Objekttyp ‚Ressource' zum Objekttyp ‚Allokation' im konzeptionellen Datenmodell zu implementieren. Dadurch kann abgebildet werden, welche Ausführungsvariante eines Vorgangs welche Ressourcen beansprucht. Die Verbindung mit dem Objekttyp ‚Allokation' liefert dabei die Höhe der jeweiligen Beanspruchung. Beziehungen mit dem Objekttyp ‚Allokation' können sowohl ausgehend vom Objekttyp ‚Modus' als auch vom Objekttyp ‚Ressource' mehrere oder auch keine eingegangen werden, da ein Modus im Zweifel mehrere Ressourcen beansprucht und eine Ressource von mehreren Modi ausgeschöpft werden kann. Objekte des Objekttyps ‚Allokation' hingegen müssen jeweils eine Verbindung mit Objekten der Objekttypen ‚Ressource' und ‚Modus' eingehen, da eine Allokation nur unter Beteiligung beider Seiten stattfindet.

5 Auswahl des Datenbankmanagementsystems

In diesem Kapitel werden zunächst verschiedene Datenbankmodelle vorgestellt. Anschließend wird eines der vorgestellten Datenbankmodelle ausgesucht, das als Grundlage für die Erfahrungsdatenbank im kerntechnischen Rückbau dienen soll. Im Anschluss daran wird ein DBMS auf Basis des ausgewählten Datenbankmodells ausgewählt, mit Hilfe dessen die Erfahrungsdatenbank implementiert und betrieben werden soll.

5.1 Datenbankmodelle

5.1.1 Hierarchisches Datenbankmodell

Das hierarchische Datenbankmodell wurde im Zuge des in den 1960er Jahren stattfindenden Apollo-Raumfahrtprogramms entwickelt. Hierbei wurden Informationen von Millionen von Einzelteilen in riesigen Datensystemen gespeichert, welche durch eine redundante Speicherung von über 60% der Informationen sehr ineffizient waren. Daher wurde die Entwicklung des Datenbanksystems ‚Information Management System' (IMS) von IBM angestoßen (Geisler, 2014, S. 52f.).

Die grundlegende Idee hinter diesem hierarchischen Datenbankmodell bestand darin, größere Baugruppen durch die zu deren Aufbau benötigten Fertigungseinheiten darzustellen, d.h. eine Stückliste zu erstellen (Geisler, 2014, S. 53). Ein denkbares, stark vereinfachtes Beispiel für den Aufbau eines hierarchischen Modells wird in Abbildung 14 ersichtlich.

Der Aufbau gleicht dem einer Baumstruktur mit einem Wurzelknoten, beliebig vielen Blattknoten sowie Eltern- und Kindknoten. Kindknoten sind untergeordnete Knoten, die im hierarchischen Modell genau einem Elternknoten untergeordnet sind. Elternknoten wiederum sind übergeordnete Knoten, denen beliebig viele Kindknoten untergeordnet sein können. Blattknoten sind Knoten, ohne weitere Kindknoten. Sie bilden das Ende eines Asts in der Baumstruktur. Besonders ist ebenfalls der Wurzelknoten, welcher der einzige Knoten der Struktur ist, der keinen Elternknoten hat. (Geisler, 2014, S. 54)

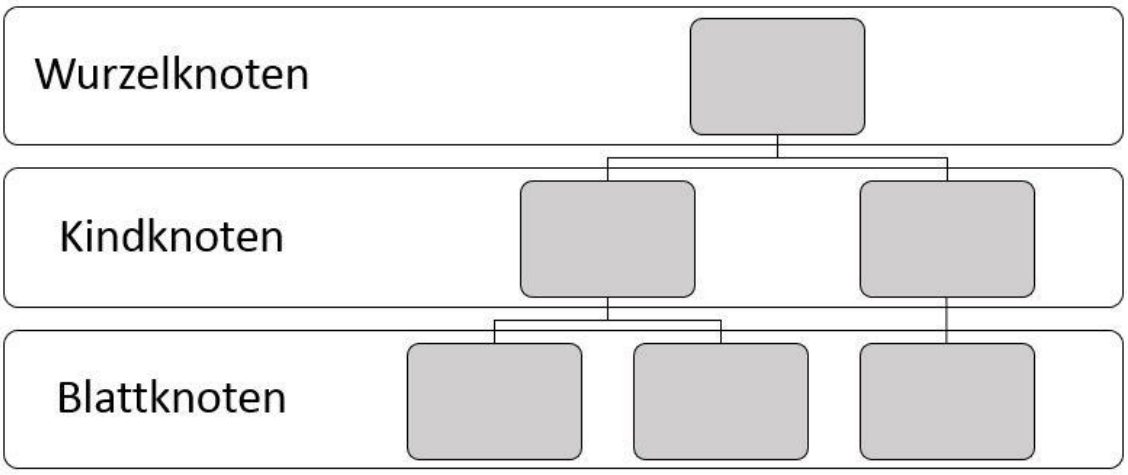


Abbildung 14: Grundkonzept des hierarchischen Modells

Eine Besonderheit des hierarchischen Datenbankmodells besteht darin, dass nur ein einziger Einstiegspunkt in eine hierarchische Datenbank gewählt werden kann und jede Informationssuche von diesem Objekttyp ausgehen muss (Jarosch, 2016, S. 117). Dieser Einstiegspunkt einer Informationssuche ist der Wurzelknoten der Baumstruktur.

Des Weiteren gibt es strukturelle Beschränkungen, welche die möglichen Verknüpfungen der Objekttypen (Knoten) untereinander betreffen. Ein übergeordneter Objekttyp kann mit keinem, einem oder mehreren untergeordneten Objekttypen in Beziehung stehen. Umgekehrt muss ein untergeordneter Objekttyp mit genau einem übergeordneten Objekttyp in Beziehung zueinander stehen. Einzige Ausnahme ist hier der Wurzelknoten, der keinen übergeordneten Knoten besitzt. Es handelt sich also um einen 1:CN-Beziehungstyp, der nur in einer Richtung, und zwar in der Hierarchie abwärts, abgegangen werden kann (Jarosch, 2016, S. 117).

Obwohl hierarchische Datenbanken teilweise bis heute noch genutzt werden, wurden sie mittlerweile größtenteils durch andere Datenbankmodelle ersetzt. Gründe für diese Verdrängung sind unter anderem dem schwierigen Management und der Implementation geschuldet. Ein weiterer Nachteil ist, dass sich hierarchische Modelle nur bedingt dafür eignen, komplexere Modelle der realen Welt abzubilden, insbesondere aufgrund ihrer Beschränkung auf 1:CN-Beziehungstypen. Kompliziertere Zusammenhänge wie etwa CM:CN-Beziehungstypen lassen sich nicht oder nur durch erheblichen Aufwand darstellen (Jarosch, 2016, S. 118f.; Geisler, 2014, S. 55f.; Schlageter und Stucky, 1983, S. 75ff.).

5.1.2 NETZWERK-DATENBANKMODELL

Um die Beschränkungen, die mit dem hierarchischen Datenmodell einhergehen, zu umgehen, wurde das sogenannte Netzwerk-Datenbankmodell entwickelt. Die

Grundlagen für das Netzwerk-Datenbankmodell bilden die auf der Conference on Data Systems Languages (CODASYL) von der Arbeitsgruppe Data Base Task Force (DBTF) erarbeiteten Vorschläge (CODASYL Data Base Task Group, 1970).

Im hierarchischen Modell ist ein Kindknoten nur in der Lage, einen einzigen Elternknoten zu haben. Im Netzwerkmodell jedoch kann ein Kindknoten mehrere Elternknoten haben. Außerdem besteht die Möglichkeit, die Beziehungen zwischen den Objekttypen in beiden Richtungen passieren zu können. Dadurch lassen sich auch CN:CM-Beziehungen abbilden, die jeweils durch eine CN:1-Beziehung der in Beziehung zu setzenden Objekttypen mit einem Koppeltyp zu realisieren sind (Jarosch, 2016, S. 121; Geisler, 2014, S. 58f.).

Durch die beiderseitige Begehbarkeit der Verbindungen zwischen den Objekttypen lässt sich der Einstieg nicht nur über einen Knoten wählen (Wurzelknoten im hierarchischen Modell), sondern „[prinzipiell] kann jeder Objekttyp als Einstiegspunkt verwendet werden“ (Jarosch, 2016, S. 119). Jedoch müssen diese Einstiegspunkte bereits beim Strukturentwurf als solche vorgesehen sein.

Die von der DBTF vorgeschlagenen Standards wurden zwar nicht einheitlich von den Datenbankherstellern übernommen, dennoch haben sie eine gegenüber dem Netzwerk-Datenbankmodell bessere Standardisierung erwirkt. Dadurch ist auch die Portabilität einer Netzwerk-Datenbank auf ein anderes DBMS einfacher zu bewerkstelligen. Allerdings muss den Anwendern beim Netzwerkmodell ebenso wie beim hierarchischen Modell die Struktur der Daten bekannt sein, um zu den gewünschten Daten navigieren zu können. Dadurch unterliegt auch das Netzwerk-Datenbankmodell einer strukturellen Abhängigkeit (Geisler, 2014, S. 59).

5.1.3 Relationales Datenbankmodell

Den Grundstein für das relationale Datenbankmodell legte der Mathematiker Codd bereits im Jahr 1970 und entwickelte dieses Modell im Laufe der Zeit beständig weiter (Jarosch, 2016, S. 125). In seiner Ausarbeitung „A Relational Model of Data for Large Shared Data Banks“ fordert Codd (1970), dass die Benutzung großer Datenbanken möglich sein muss, ohne dass bekannt ist, wie die Daten intern organisiert sind. Seiner Meinung nach sollten des Weiteren bei einer Änderung an der Struktur der Datenbank keine Anpassungen sämtlicher Anwendungsprogramme, die auf die Datenbank zugreifen, notwendig sein (Codd, 1970, 377ff.). Unter der Voraussetzung dieser Bedingung entwickelte Codd das relationale Datenbankmodell (Geisler, 2014, S. 59f.). Das relationale Datenbankmodell hat gegenüber anderen Datenmodellen den Vorteil, dass Daten miteinander in Verbindung gesetzt werden können, ohne die genauen

Zusammenhänge zum Zeitpunkt der Speicherung zu kennen. Die Möglichkeit solcher ad-hoc-Anfragen ist in der heutigen Zeit, in der sich Anforderungen binnen kürzester Zeit ändern können, von großer Bedeutung (Jarosch, 2016, S. 124). Der größte Vorteil, den ein relationales Datenbankmodell gegenüber anderen Datenbankmodellen aufweist, ist ein relationales DBMS (RDBMS), das eine komplette Abkapselung von der physikalischen Datenspeicherung ermöglicht. Somit müssen sich Anwender keine Gedanken über die physikalische Datenspeicherung machen, sondern können sich allein mit der logischen Struktur der Daten beschäftigen (Geisler, 2014, S. 60).

Aufgrund der genannten Vorteile des relationalen Datenmodells gegenüber anderen Datenmodellen, wird dieses im weiteren Verlauf dieser Studie angewendet. Aus diesem Grund wird im Folgenden detaillierter auf die Eigenschaften dieses Datenmodells eingegangen.

5.1.3.1 Grundprinzipien des relationalen Datenbank-Modells

Das Konzept der mathematischen Relation bildet die Basis für das relationale Datenmodell. Dieses ist auch als Wertetabelle beschreibbar, die ihre theoretischen Grundlagen aus der Prädikatenlogik und Mengenlehre bezieht. Die Datenbank wird dabei im relationalen Modell als Sammlung von Relationen dargestellt. Diese Relationen sind vorstellbar als Tabelle mit Werten, in der jede Zeile eine Liste zusammenhängender Datenwerte repräsentiert. Diese stellt bezugnehmend auf das ER-Modell ein Objekt bzw. eine Beziehung dar. Namen der Tabellen und Spalten geben an, wie die Bedeutung der Werte zu interpretieren ist (Elmasri und Navathe, 2011, S. 133f.).

Ein Relationsschema besteht aus dem Relationsnamen bzw. Tabellennamen und einer Liste von Attributen, die dieser zugeordnet sind. Sei ‚R' der Name einer Relation und A_1, A_2, …, A_n dessen zugeordneten Attribute, dann ist das Relationsschema folgendermaßen anzugeben: $R(A_1, A_2, ..., A_n)$ (Elmasri und Navathe, 2011, S. 135). Der Ausdehnungsgrad einer Relation gibt die Anzahl an Attributen an (Sauer, 2002, S. 33). Eine Anwendung dieser Konvention auf den Beispielobjekttyp ‚Person' mit seinen Attributen ‚Vorname', ‚Nachname', ‚Geburtsdatum' und ‚Geschlecht' ergibt das folgende Relationsschema: Person(Vorname, Nachname, Geburtsdatum, Geschlecht). Diese Relation hat folglich einen Ausdehnungsgrad von vier.

Neben den Relationen und Attributen sind auch die Wertebereiche, welche die Attribute jeweils annehmen können, von Bedeutung. Dieser Wertebereich wird auch *Domäne* genannt, welche sich für jedes Attribut in Form einer Menge angeben lässt. Der Wertebereich eines allgemein gehaltenen Attributs A_i lässt sich wie folgt

ausdrücken: Dom $[A_i] = \{a_{i1}, a_{i2}, \ldots, a_{in}\}$. In dieser Menge sind alle möglichen Attributwerte enthalten, die ein Attribut A_i als Ausprägung in einem konkreten Objekt annehmen kann. Hervorzuheben ist dabei, dass im relationalen Datenbankmodell nur atomare Werte als Attributwerte einer Domäne zugelassen sind. Ein atomarer Wert bedeutet, dass dieser nicht in sich strukturiert ist und aus Sicht des relationalen Modells unteilbar ist (Elmasri und Navathe, 2011, S. 135; Jarosch, 2016, S. 126). Beispielhaft ist in diesem Fall die Domäne des Attributs Farbe anzugeben: Dom [Farbe]={Rot, Grün, Blau}. In diesem Beispiel kann das Attribut ‚Farbe' in einem Objekt nur die Werte ‚Rot', ‚Grün' oder ‚Blau' annehmen.

Manchmal ist es jedoch nicht möglich, für jedes Attribut eines konkreten Objekts einen Wert anzugeben. Dies kann der Fall sein, wenn der Wert des Attributs für das Objekt irrelevant ist oder auch wenn der Wert des Attributs zum Zeitpunkt der Dateneingabe noch nicht bekannt ist. In diesen Fällen ist es möglich, dass ein Attribut mit einem Wert belegt wird, der genau genommen gar kein Wert ist: die sogenannte NULL-Marke. Die NULL-Marke ist nicht mit dem Zahlenwert ‚0' oder der Buchstabenfolge ‚NULL' zu verwechseln. Die NULL-Marke soll ausdrücklich darauf hinweisen, dass ein Attribut eines Objekts keinen Wert annimmt (Jarosch, 2016, S. 126).

Unter Betrachtung einer Relation als Tabelle bezeichnen die Attribute die Spalten. Zeilen hingegen können als konkrete Instanzen angesehen werden. Sie werden im relationalen Modell auch Tupel genannt. Für jedes Tupel können die Attribute jeweils einen Wert aus der jeweiligen Domäne annehmen. Jedes dieser Tupel beschreibt ein konkretes Objekt eines Objekttyps, dessen Rahmen von der Relation vorgegeben wird (Jarosch, 2016, S. 127). Die Anzahl der Tupel und damit die Anzahl der Zeilen sind grundsätzlich variabel. Jedoch sind doppelte Tupel nicht gestattet. Es kommt also zu keinem Zeitpunkt dazu, dass zwei Tupel mit den exakt gleichen Ausgestaltungen ihrer Attributwerte existieren (Sauer, 2002, S. 33).

Ähnlich zu den Objekttypen des ER-Modells aus Kapitel 4 bedarf es eines Schlüssels, um ein Tupel einer Relation eindeutig identifizieren zu können. Dies geschieht über den Primärschlüssel einer Relation. Auch hierbei kann ein natürliches Attribut zum Primärschlüssel erklärt werden oder ein künstliches Attribut wird für den Zweck der Identifizierung eingeführt. Dabei ist zu beachten, dass ein Primärschlüssel nicht mit der NULL-Marke belegt werden und in der gesamten Tabelle nur ein einziges Mal auftauchen darf, sodass eine eindeutige Identifizierung eines Tupels anhand des Primärschlüssels gewährleistet bleibt. Auch ein aus mehreren Attributen zusammengesetzter Schlüssel ist im relationalen Modell möglich (Jarosch, 2016,

S. 131ff.; Sauer, 2002, S. 34f.). Im Rahmen dieser Studie wird allerdings aus Gründen des Umfangs auf die Verwendung von zusammengesetzten Schlüsseln verzichtet.

Beziehungen zwischen Relationen werden mit Hilfe von sogenannten Fremdschlüsseln dargestellt. Sollen beispielsweise Zeilen aus einer Tabelle A jeweils auf eine Zeile von Tabelle B verweisen, d.h. es soll eine Beziehung zwischen Objekten des Typs A mit jeweils einem Objekt des Typs B hergestellt werden, dann geschieht dies durch die Aufnahme eines Fremdschlüssels. Dazu wird in unserem Beispiel der Primärschlüssel von B als zusätzliches Attribut in die Tabelle A aufgenommen. Der Primärschlüssel aus Tabelle B wird in Tabelle A als Fremdschlüssel bezeichnet. Nun enthalten die Tupel der Relation A jeweils einen Verweis auf ein Tupel der Relation B. Soll eine Zeile aus der Tabelle A nicht nur auf eine Zeile der Tabelle B verweisen, sondern zusätzlich auf Zeilen anderer Tabellen, dann müssen weitere Fremdschlüssel der jeweiligen Tabellen der Tabelle A als Attribut hinzugefügt werden (Jarosch, 2016, S. 135f.).

Die Darstellung von Primärschlüsseln und Fremdschlüsseln einer Relation R sollen im Rahmen dieser Studie durch Unterstreichung des Primärschlüssels sowie durch die Kennzeichnung der Fremdschlüssel durch die Einrahmung mit aufwärtsgerichteten Pfeilen erfolgen: R(Primärschlüssel, ↑Fremdschlüssel A↑, ↑Fremdschlüssel B↑, …, ↑Fremdschlüssel N↑, Attribut 1, Attribut 2, …, Attribut n).

Wenn Beziehungen mittels Verweistechnik per Fremdschlüssel modelliert werden, dann bestehen Einschränkungen, die beachtet werden müssen. Es kann im relationalen Modell nicht jede Art von Beziehungstyp modelliert werden, deren Modellierung im ER-Modell aus Kapitel 4.4 möglich sind. Verweisen beispielsweise Zeilen einer Tabelle A durch den ihr als Fremdschlüssel integrierten Primärschlüssel auf Zeilen der Tabelle B, dann stellt dies einen 1:CN-Beziehungstyp von B nach A dar. Da der Primärschlüssel unikal sein muss, kann der Fremdschlüssel aus A, der auf den Primärschlüssel von B verweist, nicht auf mehrere Zeilen aus B verweisen, sondern nur auf eine. Andersherum können aber mehrere Zeilen aus der Tabelle A durch ihren Fremdschlüssel auf dieselbe Zeile der Tabelle B verweisen. Daher stellt der 1:CN-Beziehungstyp im relationalen Modell den Grundtyp der repräsentierbaren Beziehungstypen dar (Jarosch, 2016, S. 139).

Ein 1:1-Beziehungstyp bedeutet, dass genau eine Zeile der einen Tabelle mit genau einer Zeile einer anderen Tabelle verknüpft ist. Dieser Zustand lässt darauf schließen, dass die Datenbank nicht mit ausreichender Sorgfalt entworfen wurde. Denn beim Vorliegen eines 1:1-Beziehungstyps ist es im Regelfall sinnvoller, die beiden involvierten Tabellen in eine zusammenzulegen (Geisler, 2014, S. 111).

Ein CN:CM-Beziehungstyp kann im relationalen Modell nicht ohne weiteres dargestellt werden. Dies liegt daran, dass als Belegung eines Attributs nur atomare Werte gestattet sind und keine Mehrfachbelegung, bzw. eine Wertemenge, möglich ist (Jarosch, 2016, S. 151). Unter der Verwendung der beiden beispielhaften Objekttypen A und B, deren Objekte jeweils mit keinem, einem oder mehreren Objekten des jeweils anderen Objekttyps in Beziehung stehen können, lässt sich dies aufzeigen. Nun müsste ein Fremdschlüsselwert einer Zeile der Tabelle A auf mehrere Zeilen der Tabelle B verweisen können, oder auch umgekehrt, um einen CN:CM-Beziehungstyp zu realisieren. Es müssten also nicht nur atomare Werte sondern auch Mehrfachbelegungen gestattet sein. Dies ist im relationalen Modell jedoch nicht möglich.

Um das Problem zu umgehen, wird eine weitere, sogenannte Koppel-Tabelle eingeführt, welche die beiden ursprünglichen Tabellen jeweils über einen 1:CN-Beziehungstyp verknüpft. Dabei muss die Koppeltabelle die Primärschlüssel der beiden ursprünglichen Tabellen jeweils als Fremdschlüssel verwenden, um einen Zusammenhang zwischen Objekten der beiden Objekttypen herzustellen (Jarosch, 2016, S. 151f.). In einem veranschaulichenden Beispiel sind die beiden Relationen Person(Ausweisnummer, Name, Geburtsdatum) und Fahrrad(Seriennummer, Farbe, Rahmengröße) mit einem CN:CM-Zusammenhang im relationalen Modell darzustellen. Eine Person kann mehrere Fahrräder benutzen und ein Fahrrad kann von mehreren Personen genutzt werden. Dazu wird eine weitere Tabelle PersonFahrrad(PersonFahrradID, ↑Ausweisnummer↑, ↑Seriennummer↑) geschaffen, die über die Verwendung der Fremdschlüssel ‚Ausweisnummer' und ‚Seriennummer' jeweils mit den Tabellen ‚Person' und ‚Fahrrad' in eine 1:CN-Beziehung gesetzt wird. In diesem Beispiel wurde in der Koppeltabelle ein organisatorischer Primärschlüssel eingefügt.

5.1.3.2 Die referenzielle Integrität

Da Beziehungen im relationalen Datenmodell auf dem Verweisprinzip beruhen, entsteht die Notwendigkeit einer Sicherstellung, dass kein Verweis ins „Leere" läuft. Es muss also sichergestellt werden, dass ein Fremdschlüssel keinen Wert annehmen kann, der nicht deckungsgleich mit einem Wert des Primärschlüssels einer anderen Tabelle ist, auf den er verweist. Dies wird im Kontext relationaler Datenbankmodelle auch als referenzielle Integrität bezeichnet (Jarosch, 2016, S. 144). Sie dient der Wahrung der Konsistenz und wird grafisch im relationalen Modell dargestellt, indem von jedem Fremdschlüssel eine Linie zu der von ihm referenzierten Relation gezeichnet wird (Elmasri und Navathe, 2011, S. 144ff.).

Ferner ist es auch möglich, die Eingabe der in Kapitel 5.1.3.1 beschriebenen NULL-Marke als Wert eines Fremdschlüssels zu gestatten. Dadurch wird dieser zu einem nicht-eingabepflichtigen Fremdschlüssel und wird in der Beschreibung der Relation neben der Einrahmung durch aufwärtsgerichteter Pfeile durch Kursivschrift gekennzeichnet (Jarosch, 2016, S. 144 ff.). Dies ist beispielsweise nötig, um einen optionalen Beziehungstyp im relationalen Datenmodell zu realisieren.

5.1.3.3 Normalisierung

Um eine qualitativ gute Tabellenstruktur im relationalen Datenmodell zu erreichen, wird sich der Normalisierung bedient. Der Prozess der Normalisierung reduziert dabei Datenredundanzen, die in der Datenbank existieren (Geisler, 2014, S. 177f.). Wie bereits in Kapitel 4.5 beschrieben wurde, existieren mehrere Normalformen, die aufeinander aufbauen. Die ersten drei Normalformen wurden dort bereits im Rahmen des konzeptionellen Modells vorgestellt. Sie sind zwar für den Bereich des relationalen Modells konzipiert, können jedoch auch im Kontext des konzeptionellen Modells Anwendung finden und so bereits für einen qualitativ hochwertigeren konzeptionellen Entwurf sorgen.

Neben den drei bereits genannten Normalformen gibt es weitere, höhere Normalformen, die jedoch in der Praxis wenig Anwendung finden. Durch den immer höher erreichten Normalisierungsgrad der Datenbank kann wiederum die Performance leiden, da das Datenmodell durch fortschreitende Normalisierung immer komplexer wird (Geisler, 2014, S. 179). Aufgrund dieser Gründe werden diese höheren Normalformen im Rahmen dieser Studie nicht näher betrachtet.

Im Rahmen dieser Studie ist jedoch die sogenannte Boyce-Codd-Normalform (BCNF) zu erwähnen, die einen Spezialfall der dritten Normalform darstellt. Eine Tabelle befindet sich in der BCNF, wenn sie in der dritten Normalform vorliegt und jede Determinante ein Schlüssel ist. Eine Determinante ist als Attribut zu verstehen, von dem andere Attribute funktional abhängig sind. Funktionale Abhängigkeit wurde bereits in Kapitel 4.5.2 erläutert. Somit befinden sich Tabellen mit einem einzigen Primärschlüssel, die also keinen zusammengesetzten Schlüssel haben und die sich in der dritten Normalform befinden, automatisch auch in der BCNF (Geisler, 2014, S. 191f.).

5.1.3.4 Structured Query Language

Relationale Datenbanken bedienen sich der relationalen Datenbanksprache Structured Query Language (SQL). Diese ist international standardisiert und wird von allen

gängigen relationalen Datenbanksystemen unterstützt (Saake et al., 2013, S. 209). Dabei kann SQL grundsätzlich in zwei verschiedenen Kategorien unterschieden werden. Zum einen tritt sie als Data Manipulation Language (DML) und zum anderen als Data Definition Language (DDL) auf.

Mit SQL lassen sich im Rahmen der DDL Datenobjekte grundsätzlich erzeugen, löschen und ändern. Im Bereich DML lassen sich Operationen durchführen, um beispielsweise Daten zu lesen, zu schreiben und zu ändern. Zu den möglichen Befehlen der Sprache, die verschiedenste Operationen beschreiben, soll im Rahmen dieser Studie jedoch nicht näher eingegangen werden, da eine detaillierte Aufgliederung der möglichen Operationen und Befehle von SQL zu umfangreich werden würde. Detaillierte Informationen zu SQL sind u.a. in Saake et al. (2013, S. 209ff.) zu finden.

5.1.4 NoSQL Datenbanken

NoSQL-Datenbanken sind vor allem im Hinblick auf die Verarbeitung sehr großer Datenmengen im Terabyte- oder sogar Petabyte-Bereich von Interesse. Mit Hilfe dieser Datenbanken kann beispielsweise Google mit den Ansätzen Map/Reduce und seinem BigTable-Datenbanksystem als einer der NoSQL-Vorreiter bezeichnet werden (Edlich et al., 2010, S. 1f.). Nicht-relationale Datenbanken in Form von hierarchischen- oder Netzwerk-Datenbanken (siehe Kapitel 5.1.1 und 5.1.2) gab es bereits vor Codds Ideen einer relationalen Datenbank. Jedoch wurden diese von relationalen Lösungen verdrängt. Erneute Bedeutung erlangten nicht-relationale Datenkonzepte mit dem Aufkommen des Internets und vielen webbasierten Anwendungen. Besonders im Big-Data-Bereich ist es teilweise unmöglich, Aufgaben mit dem relationalen Datenbankansatz zu bewältigen (Meier und Kaufmann, 2016, S. 18).

Mit dem Ausdruck *Big Data* werden Datenbestände bezeichnet, die sehr umfangreich sind sowie meistens unstrukturiert vorliegen. Des Weiteren werden die Daten aus den unterschiedlichsten Quellen bezogen. Beispiele für diese Quellen sind: E-Mails, Dokumentensammlungen, Satellitenbilder, Sensordaten und Mitteilungen aus sozialen Netzwerken. Damit sind zwei von drei Hauptcharakteristiken von Big Data genannt. Zum umfangreichen Datenbestand (Volume) und der Vielfalt von Datenformaten (Variety), beispielsweise strukturiert oder unstrukturiert, kommt noch die hohe Verarbeitungsgeschwindigkeit der Daten (Velocity) (Meier und Kaufmann, 2016, S. 11f.).

In den meisten Fällen wird unter einem NoSQL-Datenbanksystem ein System verstanden, das auf einem nicht-relationalen Datenbankmodell beruht, auf verteilte und horizontale Skalierbarkeit ausgerichtet sowie schemafrei ist (Edlich et al., 2010, S. 2).

NoSQL-Datenbanken gibt es je nach Typ in verschiedenen Ausführungsformen. Beispiele nach denen NoSQL-Systeme kategorisiert werden können, sind Key-Value Stores (Schlüssel-Wert-Datenbank), Column Stores (spaltenorientierte Datenbank), Grafendatenbanken oder auch Dokumentenspeicher (Meier und Kaufmann, 2016, S. 18f.).

5.2 Auswahl des Datenbankmodells

In diesem Abschnitt erfolgt die Auswahl eines der in den vorigen Abschnitten kurz vorgestellten Datenbankmodelle. Dazu werden verschiedene Gesichtspunkte herangezogen. Sowohl das hierarchische als auch das Netzwerk-Datenbankmodell, welche in den Kapiteln 5.1.1 und 5.1.2 beschrieben wurden, sind mittlerweile nicht mehr Stand der Technik. Sie werden heutzutage nur noch in Einzelfällen benutzt, wenn es die spezifischen Anforderungen und die Komplexität der in der Datenbank abzubildenden Realität erlauben.

Entgegen dem Nischendasein nicht-relationaler Lösungen haben sich relationale Datenbanken in den letzten Jahrzehnten zum Standard in Datenhaltungsfragen entwickelt (Piepmeyer, 2011, S. 359). In vielen Unternehmen sind sie fest etabliert und die Mitarbeiter kennen sich mit deren Umgang aus. Dies ist auch aus Abbildung 15 ersichtlich, welche die Popularität verschiedener DBMS zeigt. Relationale DBMS erfreuen sich größter Beliebtheit und haben einen beträchtlichen Vorsprung vor anderen DBMS. Mit Ausnahme der Time Series DBMS und RDBMS können alle weiter aufgeführten DBMS als Untergruppe von NoSQL-Implementierungen klassifiziert werden (DB-Engines.com, o.J.). Mit fast 80% sind relationale DBMS gegenüber NoSQL Anwendungen viel populärer.

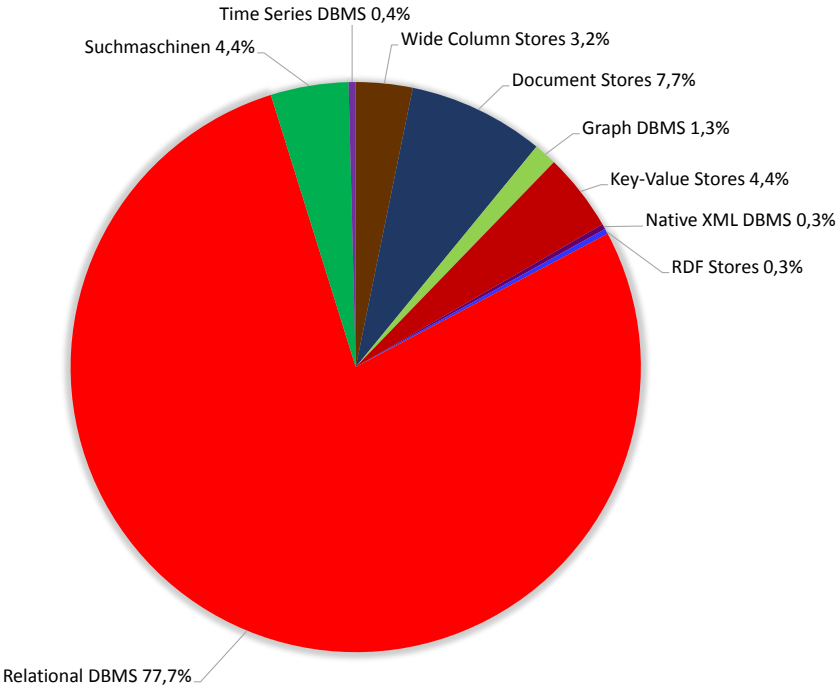


Abbildung 15: Popularität verschiedener DBMS (Quelle: DB-Engines.com, 2018b)

Diese Popularität und die damit verbundene weite Verbreitung bringen viele Vorteile mit sich. Software-Entwickler und Endnutzer sind bereits mit der Erstellung, Verwaltung und Nutzung von relationalen Datenbanken vertraut. Etablierte RDBMS haben sich bereits oft bewiesen und weisen einen höheren Reifegrad auf als noch relativ neue NoSQL-Systeme. Daher können RDBMS als zuverlässiger angesehen werden (Piepmeyer, 2011, S. 369).

Neben der Betrachtung von allgemeinen Statistiken und Nutzungsdaten ist jedoch auch der Blick auf den spezifischen Anwendungsfall lohnenswert. Im bisherigen Verlauf der Entwicklung des Expertensystems für Rückbauprojekte kerntechnischer Anlagen wird vor allen Dingen mit strukturierten Daten gearbeitet. Eine NoSQL-Anwendung, die auch die Speicherung und Verarbeitung von unstrukturierten Daten ermöglicht, wäre im Anwendungsfall dieser Studie folglich überflüssig.

Daher wird zur weiteren Entwicklung der Erfahrungsdatenbank des Expertensystems im Rahmen dieser Studie das relationale Datenbankmodell verwendet.

5.3 Auswahl des relationalen Datenbankmanagementsystems

Nach der Einschränkung auf relationale Datenbankmodelle stellt sich in diesem Abschnitt die Frage, mit welchem RDBMS das Expertensystem im Anwendungsfall dieser Studie implementiert und betrieben werden soll. Erfolgt eine Einschränkung auf die laut dem Portal DB-Engines.com im April 2018 populärsten RDBMS (DB-Engines.com, 2018a), dann bleibt eine überschaubare Anzahl an zu betrachtenden RDBMS übrig. Diese sind unter anderem MySQL, Microsoft Server SQL und Microsoft Access.

Alle drei genannten Systeme laufen auf Windows-Betriebssystemen und unterstützen eine Reihe von Programmiersprachen, wie beispielsweise C++ und Java. Des Weiteren unterstützen alle drei Systeme SQL als DDL und DML. Microsoft SQL Server kann nicht nur mit Windows-Computern, sondern auch mit Linux-Betriebssystemen genutzt werden. MySQL kann darüber hinaus mit weiteren Betriebssystemen genutzt werden. MySQL und Microsoft SQL Server sind auch für größere Datenbanken mit einer Vielzahl von gleichzeitigen Nutzern und großen zu verarbeitenden Datenmengen geeignet. Microsoft Access hingegen ist eher für kleinere Anwendungen die richtige Wahl.

Während Microsoft SQL Server und Microsoft Access kommerziell von Microsoft vertrieben werden, gibt es MySQL von Oracle als Open Source Version (DB-Engines.com, 2018c). Besonderer Wert sollte jedoch darauf gelegt werden, welcher Personenkreis später mit der Datenbank in Berührung kommt und wie die Bedienung der Datenbank für diesen ist. Diesbezüglich ist Microsoft Access den beiden anderen RDBMS vorzuziehen. Im Gegensatz zu Microsoft SQL Server und MySQL stellt Microsoft Access neben dem Backend zusätzlich ein grafisches Frontend zur Verfügung, über das die Datenabfrage bzw. Datenmanipulation benutzerfreundlich gestaltet werden kann (DB-Engines.com, 2018c). Für Personen, die keine Datenbankexperten sind, kann dadurch eine benutzerfreundliche Bedienung der Datenbank ermöglicht werden. Ein weiterer Vorteil von Microsoft Access ist die durch die Eingliederung in die Microsoft Office Suite erreichte weite Verbreitung und Kompatibilität mit anderen gängigen Programmen der Microsoft Office Suite, wie beispielsweise Microsoft Excel.

Im Rahmen dieser Studie wird nur eine vergleichsweise kleine relationale Datenbank benötigt und daher ist Microsoft Access für diesen Zweck ausreichend. Sollte die Datenbankanwendung dennoch im Zeitverlauf zu einer komplexeren und

umfangreicheren heranwachsen, dann ist jederzeit ein Upsizing durch die Migration auf eine Microsoft SQL Server-Datenbank möglich (Microsoft Corporation, o.J).

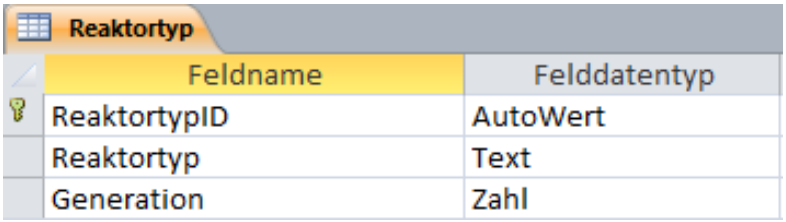
Aus diesem Grund ist es sinnvoll, dass am Anfang des Lebenszyklus das im Rahmen dieser Studie erarbeitete Expertensystem in Microsoft Access implementiert wird. Dadurch ist gleichzeitig die Benutzerfreundlichkeit durch das grafische Frontend gewährleistet, ohne eine spätere Skalierbarkeit und Migration in ein größeres System auszuschließen.

6 Logisches Design

In diesem Kapitel wird das plattformunabhängige konzeptionelle Datenmodell aus Kapitel 4 auf das in Kapitel 5 ausgewählte Datenbankmodell angepasst und es wird mit Hilfe des in Kapitel 5 ausgewählten DBMS implementiert. Das Zieldatenbankmodell ist demnach das relationale Datenmodell, welches mit Hilfe des RDBMS Microsoft Access umgesetzt wird.

Beim Vergleich der Konventionen des ER-Modells aus Kapitel 4 mit denen des relationalen Datenmodells aus Kapitel 5.1.3 werden einige Anpassungen des konzeptionellen Schemas offensichtlich, um es zu einem relationalen Modell umzuwandeln. In einem ersten Schritt sind die Objekttypen mit ihren Attributen in Relationen umzuwandeln, die in Tabellenform dargestellt werden können. Der zweite Schritt betrifft die Beziehungstypen, die mit Hilfe des Verweisprinzips mittels Fremdschlüsseln in die Relationen integriert werden. Dabei ist insbesondere auf die Art der Beziehungstypen zu achten, da nicht alle Klassen von Beziehungstypen im relationalen Modell darstellbar sind. Anstatt von Objekttypen und Beziehungstypen im konzeptionellen ER-Modell existieren im logischen Entwurf des relationalen Modells nur noch Relationen, die auch als Tabellen darstellbar sind, um eine bessere Verständlichkeit zu erzielen.

Die erste im Rahmen dieser Studie zu betrachtende Relation ist die Relation ‚Reaktortyp', welche in Abbildung 16 dargestellt ist. Als Primärschlüssel fungiert das organisatorische Attribut ‚ReaktortypID', welches durch Unterstreichung kenntlich gemacht ist. Diesem wurde mit Hilfe von Microsoft Access der Felddatentyp ‚AutoWert' zugewiesen. Somit beinhaltet die Domäne dieses Attributs alle Zahlen vom Zahlentyp ‚Long Integer' und der Wert wird automatisch beim Anlegen einer neuen Instanz mit einer fortlaufenden Zahl aus diesem Bereich belegt.



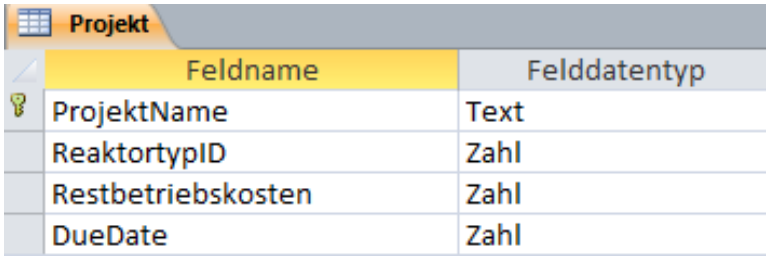
Reaktortyp(<u>ReaktortypID</u>, Reaktortyp, Generation)

Abbildung 16: Relation ‚Reaktortyp'

Weitere Attribute sind wie im konzeptionellen Modell aus Kapitel 4.6 ‚Reaktortyp ' und ‚Generation'. Diese können mit einem kurzen Text bis zu 100 Zeichen bzw. mit einer Zahl vom Typ Integer belegt werden. Die Eingabe eines Wertes beim Attribut ‚Reaktortyp' ist im Anwendungsfall obligatorisch, während nicht zwingend eine Generation hinterlegt werden muss. Dies ist dem Umstand geschuldet, dass es nicht zu jedem Reaktortyp unterschiedliche Generationen gibt.

Die Relation ‚Projekt' ist in Abbildung 17 dargestellt. Als eindeutiger Primärschlüssel dient hierbei das Attribut ‚ProjektName', welches mit einem kurzen Text von maximal 100 Zeichen zu belegen ist. Als Fremdschlüssel ist in dieser Relation der Primärschlüssel ‚ReaktortypID' aus der Relation ‚Reaktortyp' beinhaltet, der mit einem Wert aus der Domäne des Primärschlüssels belegt werden muss. Die Eingabe des Fremdschlüssels ist erforderlich, sodass ein 1:CN-Beziehungstyp von der Relation ‚Reaktortyp' zur Relation ‚Projekt' entsteht.

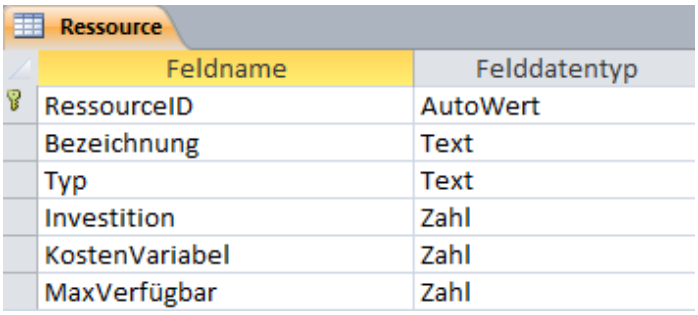
Die beiden verbleibenden Attribute ‚Restbetriebskosten' und ‚DueDate' sind beide mit einem Zahlenwert zu belegen und die Eingabe eines Wertes ist bei beiden genannten Attributen ebenfalls obligatorisch. Die Hinterlegung eines Zahlenwerts beim Attribut ‚Restbetriebskosten' ist in Form des Typs Double möglich, während beim Attribut ‚DueDate' nur Zahlenwerte vom Typ Long Integer speicherbar sind.



Projekt(<u>ProjektName</u>, ↑ReaktortypID↑, Restbetriebskosten, DueDate)

Abbildung 17: Relation ‚Projekt'

Die zur Ausführung eines Projektes benötigten Ressourcen werden im relationalen Modell in der Relation ‚Ressource' modelliert. Wie in Abbildung 18 zu sehen ist, wird als Primärschlüssel das Attribut ‚RessourceID' verwendet, das automatisch mit einem Wert vom Typ Long Integer belegt wird. Das Attribut ‚Bezeichnung' muss mit einem kurzen Text bis zu einer maximalen Länge von 255 Zeichen belegt werden. Das Attribut ‚Typ' hingegen muss mit einem kurzen Text von maximal 50 Zeichen belegt werden. Auch für die beiden Attribute ‚Investition', ‚KostenVariabel' und ‚MaxVerfügbar' ist die Eingabe eines Wertes erforderlich. Hier können jedoch jeweils Zahlenwerte vom Typ Double hinterlegt werden.



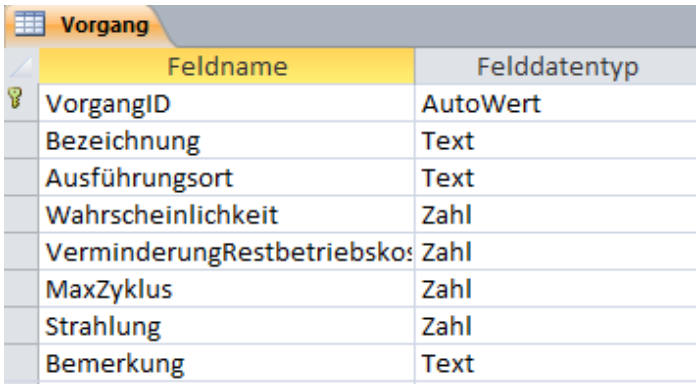
Ressource

Feldname	Felddatentyp
RessourceID	AutoWert
Bezeichnung	Text
Typ	Text
Investition	Zahl
KostenVariabel	Zahl
MaxVerfügbar	Zahl

Ressource(RessourceID, Bezeichnung, Typ, Investition, KostenVariabel, MaxVerfügbar)

Abbildung 18: Relation ‚Ressource'

Die Relation ‚Vorgang', die in Abbildung 19 dargestellt ist, besteht aus acht Attributen. Als Primärschlüssel fungiert das Attribut ‚VorgangID' welches automatisch beim Anlegen eines neuen Datensatzes mit einem eindeutigen Wert vom Typ Long Integer belegt wird. Die drei Attribute ‚Bezeichnung', ‚Ausführungsort' und ‚Bemerkung' sind jeweils mit einem kurzen Text mit einer maximalen Zeichenlänge von 255 zu belegen. Die Hinterlegung eines Wertes ist dabei für die Attribute ‚Ausführungsort' und ‚Bemerkung' optional, für das Attribut ‚Bezeichnung' jedoch obligatorisch. Die Attribute ‚Wahrscheinlichkeit' und ‚VerminderungRestbetriebskosten' müssen mit einem Zahlenwert vom Typ Double belegt werden. Die Maximale Zyklusanzahl eines Vorgangs wird im Attribut ‚MaxZyklus' in Form eines Zahlenwertes vom Typ Integer hinterlegt, wobei der Standardwert auf 1 gesetzt ist. Das Attribut ‚Strahlung' gibt die Strahlungsintensität wieder und ist abseits des Standardwerts von 0 mit einem Zahlenwert vom Typ Double zu belegen. An dieser Stelle ist die Annahme zu nennen, dass die Zykluszahl sowie die Strahlungsintensität für Mitarbeiter bei der Ausführung des Vorgangs unabhängig vom Modus sind.



Vorgang

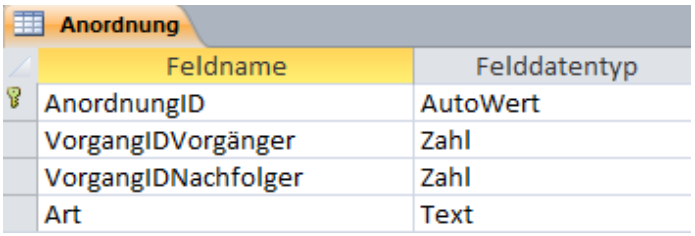
Feldname	Felddatentyp
VorgangID	AutoWert
Bezeichnung	Text
Ausführungsort	Text
Wahrscheinlichkeit	Zahl
VerminderungRestbetriebsko	Zahl
MaxZyklus	Zahl
Strahlung	Zahl
Bemerkung	Text

Vorgang(VorgangID, Bezeichnung, Ausführungsort, Wahrscheinlichkeit, VerminderungRestbetriebskosten, MaxZyklus, Strahlung, Bemerkung)

Abbildung 19: Relation ‚Vorgang'

Wie bereits in der Anforderungsanalyse aus Kapitel 3 und dem konzeptionellen Modell aus Kapitel 4.6 bekannt ist, müssen bei den Vorgängen auch im relationalen Modell zeitliche Anordnungsbeziehungen zwischen diesen modelliert werden. Dies geschieht über die Relation ‚Anordnung' (vgl. Abbildung 20). Als Primärschlüssel dieser Relation ist das Attribut ‚AnordnungID' zu nennen, welches ebenfalls automatisch mit einem für die Relation eindeutigen Zahlenwert vom Typ Long Integer belegt wird. Des Weiteren enthält die Relation zwei Fremdschlüsselattribute, die durch Einrahmung mit aufwärtsgerichteten Pfeilen kenntlich gemacht sind: ‚VorgangIDVorgänger' und ‚VorgangIDNachfolger'.

Beide sind über referenzielle Integrität mit dem Primärschlüssel ‚VorgangID' aus der Relation ‚Vorgang' verbunden und können somit nur Werte annehmen, die bereits dort vergeben sind. Dadurch werden jeweils ein 1:CN-Beziehungstyp vom Attribut ‚VorgangID' aus der Relation ‚Vorgang' zu den Attributen ‚VorgangIDVorgänger' und ‚VorgangIDNachfolger' der Relation ‚Anordnung' implementiert. Als letztes Attribut der Relation ‚Vorgang' ist das Attribut ‚Typ' zu nennen, welches die Art der Anordnungsbeziehung angibt (ob es sich um einen direkten oder einen normalen Nachfolger handelt).



Anordnung

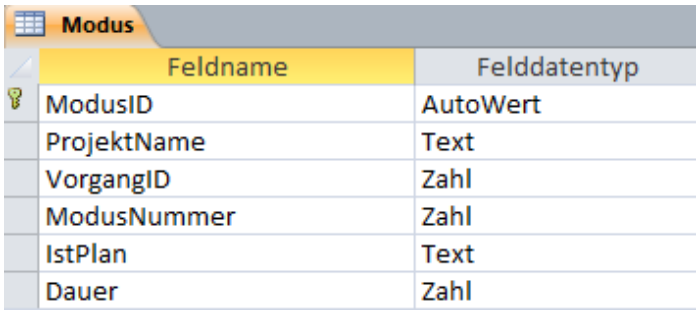
Feldname	Felddatentyp
AnordnungID	AutoWert
VorgangIDVorgänger	Zahl
VorgangIDNachfolger	Zahl
Art	Text

Anordnung(<u>AnordnungID</u>, ↑VorgangIDVorgänger↑, ↑VorgangIDNachfolger↑, Art)

Abbildung 20: Relation ‚Anordnung'

Jeder Vorgang kann in verschiedenen Modi ausgeführt werden. Die je Modus zu speichernden Informationen sind in der Relation ‚Modus' hinterlegt (vgl. Abbildung 21). Auch hierbei gibt es einen automatisch generierten Primärschlüssel, der im Attribut ‚ModusID' gespeichert wird. Zusätzlich werden mit den beiden Fremdschlüsseln ‚ProjektName' (mit referenzieller Integrität auf den Primärschlüssel der Relation ‚Projekt') und ‚VorgangID' (mit referenzieller Integrität auf den Primärschlüssel der Relation ‚Vorgang') Beziehungen mit diesen beiden genannten Relationen generiert. Es handelt sich jeweils um einen 1:CN-Beziehungstypen, der jeweils von den Relationen ‚Vorgang' und ‚Projekt' in Richtung der Relation ‚Modus' geht.

Zusätzliche Attribute der Relation sind ‚ModusNummer', ‚IstPlan' und ‚Dauer'. Das Attribut ‚ModusNummer' ist mit einem Zahlenwert vom Typ Integer oder dem hinterlegten Standardwert von 1 zu belegen. Für das Attribut ‚IstPlan' gibt es nur zwei Möglichkeiten der Wertbelegung. Entweder es wird mit dem Text ‚Ist' oder mit dem Text ‚Plan' belegt, welcher die Dimension des Wertes des Attributs ‚Dauer' angibt. Die Wertbelegung des Attributs ‚Dauer' wiederum hat mit einem Zahlenwert vom Typ Double zu erfolgen.



Modus

Feldname	Felddatentyp
ModusID	AutoWert
ProjektName	Text
VorgangID	Zahl
ModusNummer	Zahl
IstPlan	Text
Dauer	Zahl

Modus(<u>ModusID</u>, ↑ProjektName↑, ↑VorgangID↑, ModusNummer, IstPlan, Dauer)

Abbildung 21: Relation ‚Modus'

Als letzte Relation ist im logischen Entwurf dieser Studie die Relation ‚Allokation' vorzustellen (vgl. Abbildung 22). Auch hierbei wird der Primärschlüssel ‚AllokationID' automatisch mit einem Zahlenwert vom Typ Long Integer belegt. Da in dieser Relation gespeichert wird, welche Ressourcen von welchem Ausführungsmodus eines Vorgangs beansprucht werden, müssen Verbindungen mit den Relationen ‚Ressource' und ‚Modus' geschaffen werden. Dies geschieht über die beiden Fremdschlüsselattribute ‚RessourceID' und ‚ModusID', die jeweils über referenzielle Integrität mit den Primärschlüsseln der Relationen ‚Ressource' und ‚Modus' verknüpft sind. Dadurch entstehen zwei 1:CN-Beziehungstypen, die jeweils von der anderen Relation in Richtung der Relation ‚Allokation' bestehen. Die Eingabe beider Fremdschlüssel ist erforderlich. Ebenso ist die Eingabe eines Zahlenwertes vom Typ Double für das Attribut ‚Menge' erforderlich, um die Höhe der beanspruchten Ressourcen festzuhalten. Dabei ist ein Standardwert von 0 hinterlegt.

Allokation

Feldname	Felddatentyp
AllokationID	AutoWert
RessourceID	Zahl
ModusID	Zahl
Menge	Zahl

Allokation(<u>AllokationID</u>, ↑RessourceID↑, ↑ModusID↑, Menge)

Abbildung 22: Relation ‚Allokation'

Das gesamte logische Design mit allen Beziehungen der zuvor genannten Relationen ist in Abbildung 23 dargestellt.

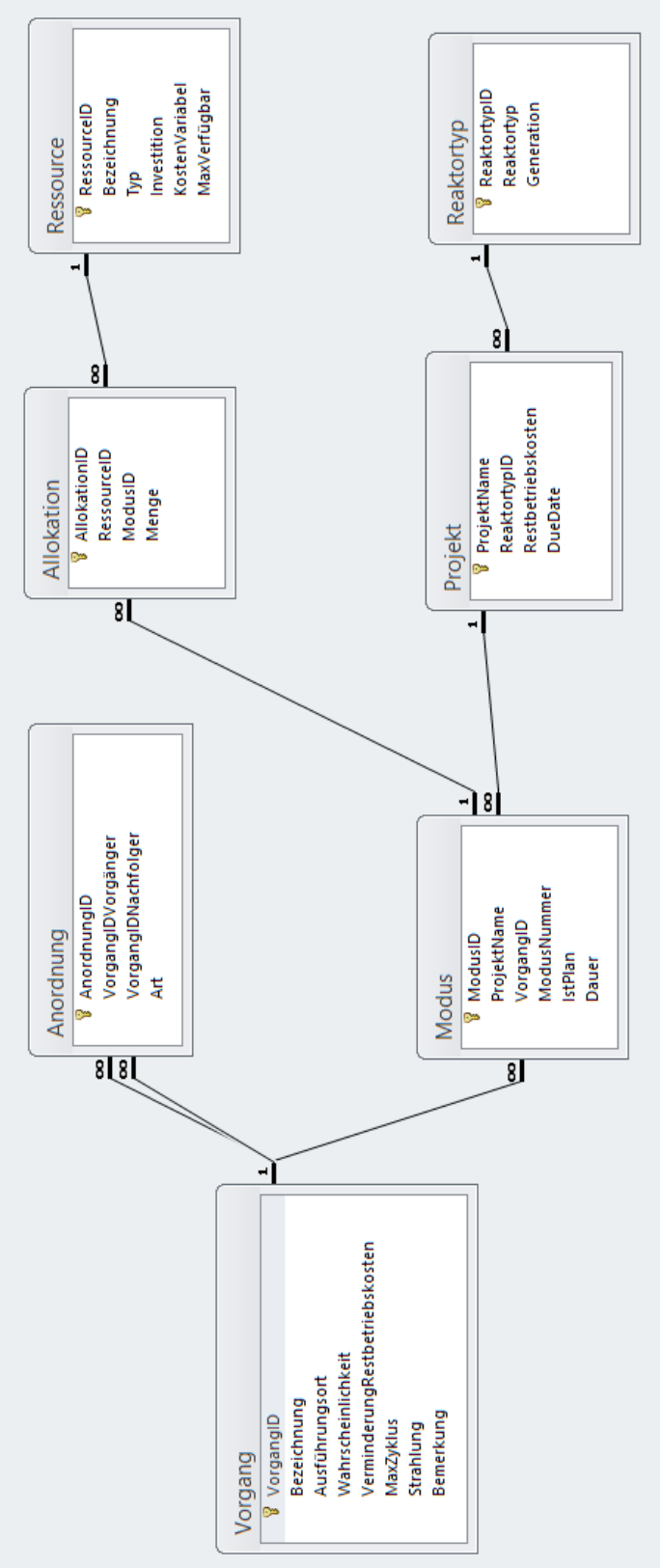


Abbildung 23: Logisches Design des Expertensystems, implementiert in Microsoft Access

7 Physikalischer Entwurf

Da im Rahmen dieser Studie ein relationales Datenbankmodell mitsamt einem RDBMS verwendet wird, ist der letzte Schritt im Entwicklungsprozess einer Datenbank hinfällig geworden. Aufgrund der resultierenden starken logischen Isolierung von relationalen Datenbanksystemen ist „die Notwendigkeit, sich mit der physikalischen Speicherung der Daten zu befassen“ (Geisler, 2014, S. 306) nicht mehr gegeben bzw. gesunken. Gemäß Geisler (2014) wird beispielsweise „durch das RDBMS eine komplette Kapselung der physikalischen Datenspeicherung erreicht“ (Geisler, 2014, S. 60).

Daher wird auch im Entwicklungsprozess des Expertensystems kerntechnischer Anlagen im Rahmen dieser Studie die Betrachtung des physikalischen Entwurfs nicht vollzogen.

8 Expertensystem im Anwendungsfall

Ausgehend vom logischen Entwurf aus Kapitel 6 wird das Expertensystem für den Rückbau kerntechnischer Anlagen mit dem in Kapitel 5.3 ausgewählten RDBMS Microsoft Access realisiert. Bevor das Expertensystem jedoch verwendet werden kann, muss darauf geachtet werden, dass im Editor für Visual Basic for Applications (VBA) die in Abbildung 24 ersichtlichen Verweise auf Objektbibliotheken gesetzt sind. Nur wenn die aufgelisteten Verweise aktiviert sind, ist eine fehlerfreie Ausführung der Datenbank möglich.

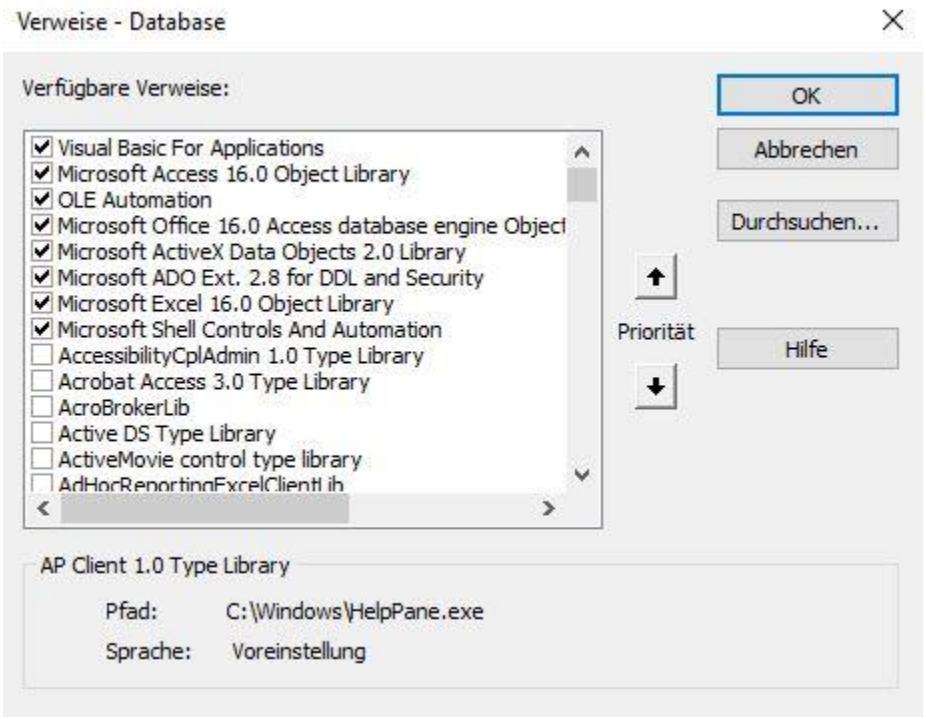


Abbildung 24: Verweise auf Objektbibliotheken in Microsoft Access

Bei der Verwendung des Expertensystems ist zwischen der Dateneingabe und der Datenabfrage zu unterscheiden. Erfahrungswerte eines abgeschlossenen Projekts können durch eine Dateneingabe in der Erfahrungsdatenbank des Expertensystems gespeichert werden. Um zu prüfen, ob relevante Daten in der Erfahrungsdatenbank des Expertensystems für ein neu zu planendes Projekt enthalten sind, können durch intelligente Abfragen entsprechende Daten extrahiert werden. In Kapitel 8.1 wird zunächst die Funktionsweise der Dateneingabe und anschließend in Kapitel 8.2 die Datenabfrage erläutert.

8.1 Dateneingabe und Änderung von gespeicherten Daten

Im Folgenden wird das Vorgehen zum Eintragen von Daten über die erstellten Formulare in Microsoft Access beschrieben.

Beim Starten des Expertensystems in Microsoft Access öffnet sich zu Beginn das in Abbildung 25 abgebildete Formular „f_Start“.

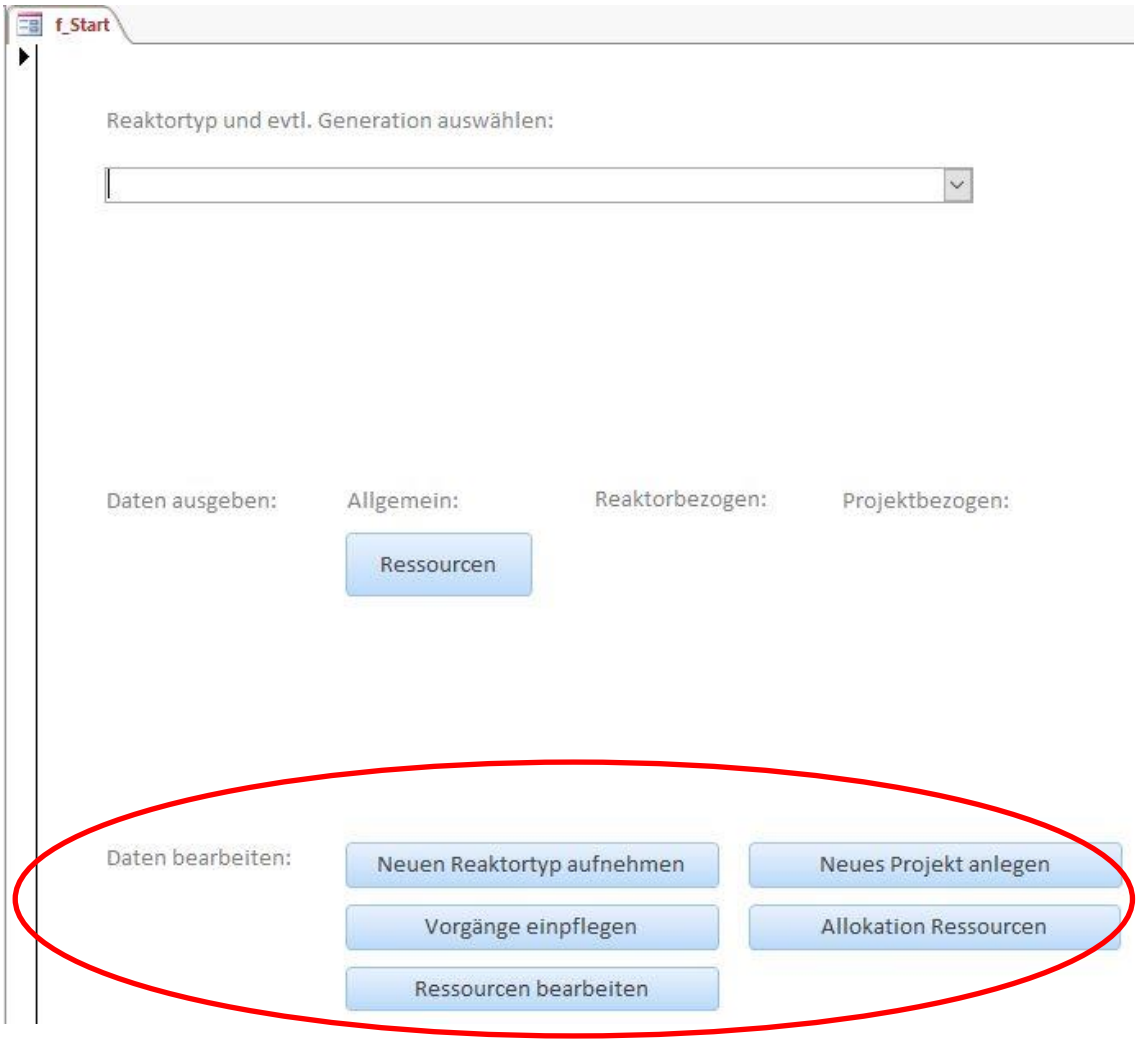


Abbildung 25: Formular "f_Start" beim Starten des Expertensystems

Zunächst ist für die Dateneingabe und Datenbearbeitung nur der untere, rot umrandete Teil des Formulars von Interesse. Dort sind fünf Buttons unter der Rubrik ‚Daten bearbeiten' zu finden. Durch das Klicken auf die Schaltfläche mit der Aufschrift ‚Neuen Reaktortyp aufnehmen' öffnet sich das Formular ‚Formular_Reaktortyp' (vgl. Abbildung 26).

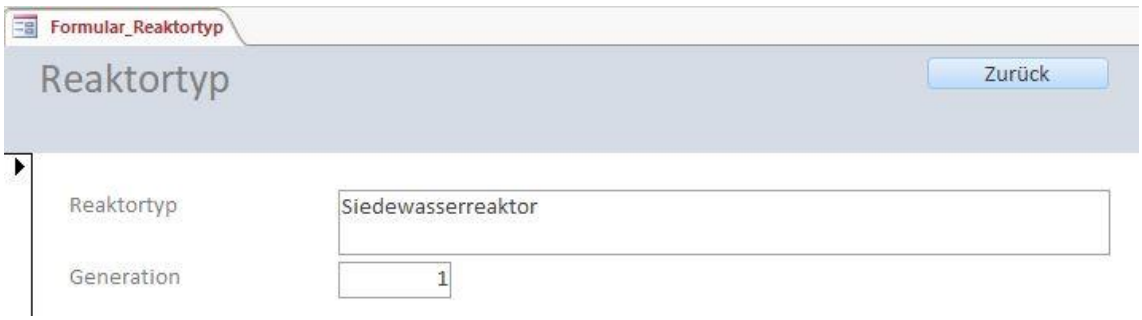


Abbildung 26: Formular ‚Formular_Reaktortyp'

In der Navigationsleiste am unteren Bildrand lassen sich die bestehenden Datensätze anschauen oder neue leere Datensätze aufrufen. Bereits bestehende Datensätze können bei Wunsch geändert werden. Wird jedoch ein neuer leerer Datensatz aufgerufen, dann sind zwei leere Eingabefelder zu erkennen. Im ersten Feld ist der Reaktortyp in Textform einzutragen und im zweiten Feld ist die Generation des Reaktortyps als ganze Zahl anzugeben. Ist bei einem speziellen Reaktortyp keine Generation anzugeben, dann kann dieses Feld auch leer gelassen werden. Nach dem Anlegen der gewünschten neuen Datensätze kann die Schaltfläche mit der Aufschrift ‚Zurück' betätigt werden, die sich oben rechts im Formular befindet. Durch das Klicken dieser Schaltfläche schließt sich das Formular ‚Formular_Reaktortyp' und der Anwender gelangt zurück zum Startformular ‚f_Start'.

Sofern ein neues Projekt angelegt werden soll, kann dies analog durch das Klicken auf die Schaltfläche mit der Aufschrift ‚Neues Projekt anlegen' im unteren Bereich des Startformulars ‚f_Start' geschehen. Dadurch öffnet sich das Formular ‚f_Projekt' (vgl. Abbildung 27).

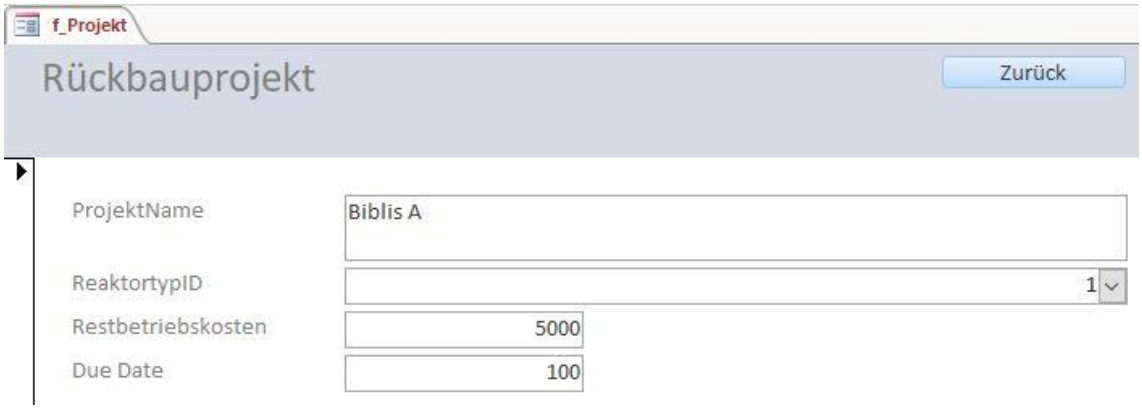


Abbildung 27: Formular ‚f_Projekt'

Auch in diesem Formular können mit Hilfe der Navigationsleiste am unteren Bildschirmrand bestehende Datensätze durchgesehen und bearbeitet sowie neue Projekte angelegt werden. Um ein neues Projekt anzulegen, sind vier Eingabefelder zu befüllen. Im ersten Feld ist der Name des neuen Projekts in Textform einzugeben. In Feld zwei findet sich ein Drop-Down-Menü, mit dessen Hilfe dem neu anzulegenden Projekt ein bereits eingepflegter Reaktortyp zugewiesen werden kann. In den beiden verbliebenen Eingabefeldern sind abschließend noch im Feld ‚Restbetriebskosten' die Restbetriebskosten zu Beginn des Rückbauprojekts und die Deadline im Feld ‚Due Date' zu hinterlegen. Sind alle gewünschten neuen Projekte mitsamt ihrer Informationen hinterlegt, dann kann analog zum vorherig genannten Eingabeformular

in der oberen rechten Ecke die Schaltfläche ‚Zurück' geklickt werden, um zur Ausgangsseite zurückzukehren.

Durch das Klicken auf die Schaltfläche ‚Vorgänge einpflegen' im unteren Bereich des Startformulars ‚f_Start' öffnet sich das Formular ‚f_Vorgänge' (vgl. Abbildung 28). Dieses Formular ist umfangreicher als die anderen bisher beschriebenen Formulare. Dies liegt daran, dass in das Formular ‚f_Vorgänge' Unterformulare eingegliedert sind, um die Anordnungsbeziehungen und Modi von Vorgängen ebenfalls mit diesem Formular abzudecken. Auch hier lassen sich durch die Navigationsleiste am unteren Rand bestehende Datensätze durchsehen bzw. abändern und erweitern sowie neue leere Datensätze mit Daten befüllen.

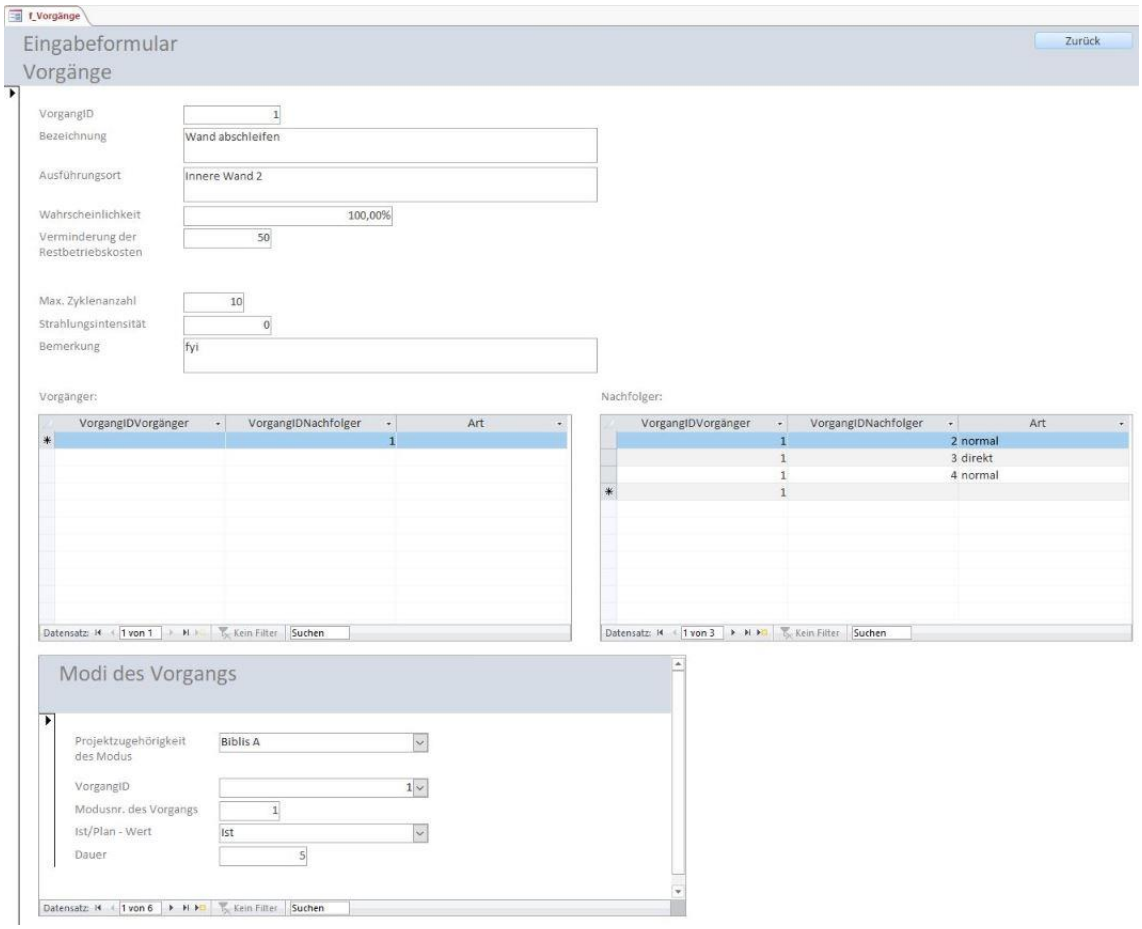


Abbildung 28: Formular ‚f_Vorgänge' mit Unterformularen

Im Feld ‚VorgangID' wird automatisch für jeden Vorgang eine eindeutige Nummer generiert. Im Feld ‚Bezeichnung' ist die Bezeichnung des Vorgangs in Textform einzutragen. Auch der Ausführungsort ist in Textform im Feld ‚Ausführungsort' anzugeben. Die Wahrscheinlichkeit der Vorgangsausführung, die Verminderung der Restbetriebskosten aufgrund der Vorgangsausführung, die maximale Anzahl an Zyklen und die Strahlungsintensität sind in Zahlenwerten einzutragen. Dabei sind jeweils

Standardwerte von 0 hinterlegt, außer bei der maximalen Anzahl an Zyklen, die als Standardwert 1 hinterlegt hat. Optional ist dem jeweiligen Vorgang eine Bemerkung hinzuzufügen.

Über die beiden Unterformulare ‚Vorgänger' und ‚Nachfolger' können alle Vorgänger und Nachfolger des aktuell einzupflegenden Vorgangs über jeweils ein Drop-Down-Menü ausgewählt werden. Hierzu ist zum Eintragen aller Vorgänger im linken Unterformular, das mit ‚Vorgänger' betitelt ist, jeweils der aktuelle Vorgang unter ‚VorgangIDNachfolger' einzutragen und alle Vorgänger dieses Vorgangs sind unter der Rubrik ‚VorgangIDVorgänger' einzupflegen bzw. einzusehen. Sofern Nachfolger des aktuell betrachteten Vorgangs eingetragen werden sollen, können im rechten Unterformular, welches mit ‚Nachfolger' betitelt ist, jeweils der aktuelle Vorgang unter ‚VorgangIDVorgänger' eingetragen und alle Nachfolger dieses Vorgangs unter der Rubrik ‚VorgangIDNachfolger' eingepflegt bzw. eingesehen werden. In beiden Unterformularen ist des Weiteren die Art der Anordnungsbeziehung zu erkennen bzw. auszuwählen. Hierbei ist zwischen den Varianten ‚normal' und ‚direkt' zu wählen. Werden zukünftig weitere Arten an Anordnungsbeziehungen benötigt, lassen sich diese jederzeit über die Entwurfsansicht in das Auswahlmenü integrieren.

Im letzten Abschnitt im Eingabeformular der Vorgänge findet sich das Unterformular, in dem die Ausführungsmodi eines Vorgangs gespeichert werden. Im ersten Auswahlfeld ist der Modus eines Vorgangs durch die Drop-Down-Liste einem eingepflegten Projekt zuzuordnen. Das zweite Feld wird automatisch mit der ID des aktuellen Vorgangs befüllt. Anschließend ist noch einzutragen, um welche Modusnummer des Vorgangs es sich handelt und ob es sich bei der einzutragenden Dauer um einen Ist- oder einen Planwert handelt.

Durch das Klicken auf die Schaltfläche mit der Aufschrift ‚Zurück' im oberen Bereich des Formulars ‚f_Vorgänge' wird dieses geschlossen und das Startformular ‚f_Start' wird erneut zur aktiven Ansicht geöffnet.

Es befinden sich zwei weitere Schaltflächen im unteren Bereich des Startformulars ‚f_Start'. Durch das Klicken auf die Schaltfläche mit der Aufschrift ‚Ressourcen bearbeiten' im unteren Bereich des Startformulars ‚f_Start' wird das Formular ‚f_Ressource' geöffnet. Dieses Formular funktioniert analog zu den anderen Formularen. Im ersten Feld wird, wie in Abbildung 29 dargestellt, die automatisch generierte ‚RessourceID' angezeigt. Des Weiteren sind die Bezeichnung der Ressource in Textform und Investitionskosten sowie variable Kosten in Zahlenform anzugeben. Der Typ der Ressource kann über ein Drop-Down-Menü ausgewählt

werden. Momentan ist in diesem die Auswahl zwischen ‚erneuerbar' und ‚nicht erneuerbar' zu treffen. Diese Auswahl lässt sich jedoch jederzeit über die Entwurfsansicht des Formulars erweitern.

Abbildung 29: Formular ‚f_Ressource'

Durch das Klicken auf die Schaltfläche mit der Aufschrift ‚Allokation Ressourcen' öffnet sich das Formular ‚f_Allokation' (vgl. Abbildung 30). Dieses Formular hat zwei Drop-Down-Menüs. Zum einen kann je Modus die benötigte Ressource ausgewählt werden. Zum anderen kann der Modus in Abhängigkeit des Projekts und des Vorgangs ausgewählt werden, der die Ressource beansprucht. Die Anzahl an benötigten Ressourcen ist als Zahlenwert im letzten Feld einzutragen. Analog zu den anderen Formularen befindet sich auch in diesem Formular die Schaltfläche mit der Aufschrift ‚Zurück' in der rechten oberen Ecke.

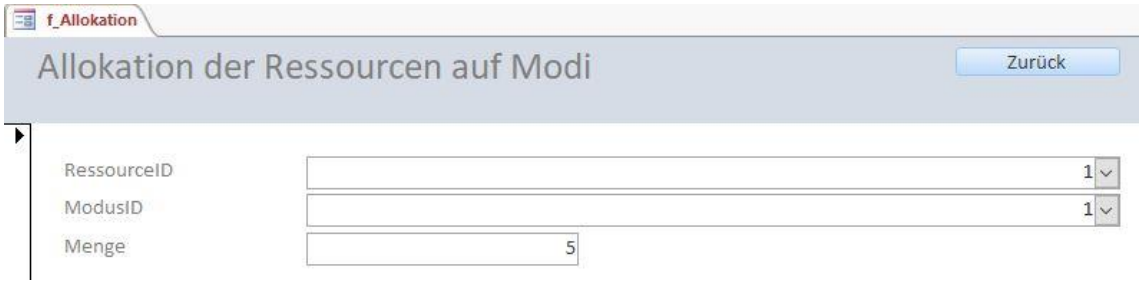


Abbildung 30: Formular ‚f_Allokation'

8.2 Datenabfrage

Analog zur Dateneingabe und Datenbearbeitung ist der Ausgangspunkt zur Datenabfrage ebenfalls das in Abbildung 25 gezeigte Formular ‚f_Start'. Im Gegensatz zur Dateneingabe und Datenänderung ist jedoch nicht mehr der untere Teil des Formulars von Interesse, sondern das Drop-Down-Menü im oberen Bereich des Formulars, welches dazu auffordert, einen Reaktortyp und dessen Generation

auszuwählen. Darüber hinaus sind die Schaltflächen unter der Rubrik ‚Daten ausgeben:' relevant. In diesem Bereich ist zunächst nur die Schaltfläche ‚Ressourcen' unter der Spaltenüberschrift ‚Allgemein:' sichtbar. Eine Betätigung dieser Schaltfläche öffnet den Bericht ‚b_Ressourcen', der alle in der Erfahrungsdatenbank enthaltenen Ressourcen mitsamt ihren Investitionskosten, variablen Kosten und ihrem Typ auflistet.

Sofern im obersten Drop-Down-Menü des Startformulars ‚f_Start' ein Reaktortyp und gegebenenfalls dessen Generation ausgewählt wurden, erscheinen weitere Schaltflächen unter der Rubrik ‚Reaktorbezogen:' und ein zweites Drop-Down-Menü mit der Aufforderung ein Rückbauprojekt auszuwählen (vgl. Abbildung 31).

Durch das Klicken auf die Schaltfläche mit der Beschriftung ‚Durchlauf Reaktortyp' öffnet sich ein Bericht, der alle Vorgänge ausgibt, die Teil eines Rückbauprojekts des gewählten Reaktortyps (und evtl. der gewählten Generation) sind. Den Vorgängen werden dabei die Anzahl ihrer ausführbaren Modi, ihre Ausführungswahrscheinlichkeit, die Verminderung der Restbetriebskosten sowie die maximale Anzahl an durchführbaren Zyklen zugeordnet. Des Weiteren wird ersichtlich, welche Vorgänger ein Vorgang hat und die Anzahl sowie die Auflistung der normalen als auch der direkten Nachfolger wird ebenfalls für jeden Vorgang ausgegeben.

Die Betätigung der zweiten neu erschienene Schaltfläche mit der Aufschrift ‚Dauer Reaktortyp' öffnet den Bericht ‚b_Dauer_Reaktorbezogen'. In diesem werden ebenfalls alle Vorgänge aufgeführt, die Teil eines Rückbauprojekts des gewählten Reaktortyps (und evtl. der gewählten Generation) sind. Bei diesem Bericht liefert die Ausgabe zu jedem Vorgang die Modusnummern, in welchen dieser ausgeführt wurde oder werden soll, sowie die Dauer und deren Art der Verteilung. Des Weiteren werden die Art und die Menge der benötigten Ressourcen ausgegeben. Ist die Dauer eines Modus eines Vorgangs in allen in der Erfahrungsdatenbank hinterlegten Projekten des ausgewählten Reaktortyps (und der gewählten Generation) dieselbe, dann wird die Art der Verteilung mit einem ‚DET' gekennzeichnet, was einer deterministischen Verteilung entspricht. Existieren unterschiedliche Dauern des gewählten Vorgangs in verschiedenen Projekten des ausgewählten Reaktortyps (und der gewählten Generation), dann wird die Dauer in folgendem Format ausgegeben: Als erster Wert wird die minimale Dauer, als zweiter Wert die maximale Dauer und als dritter Wert die durchschnittliche Dauer über alle Projekte hinweg angegeben. Diese drei Werte werden jeweils durch ein Semikolon getrennt. In diesem Fall liegt eine Betaverteilung vor, die mit ‚BETAS' gekennzeichnet wird.

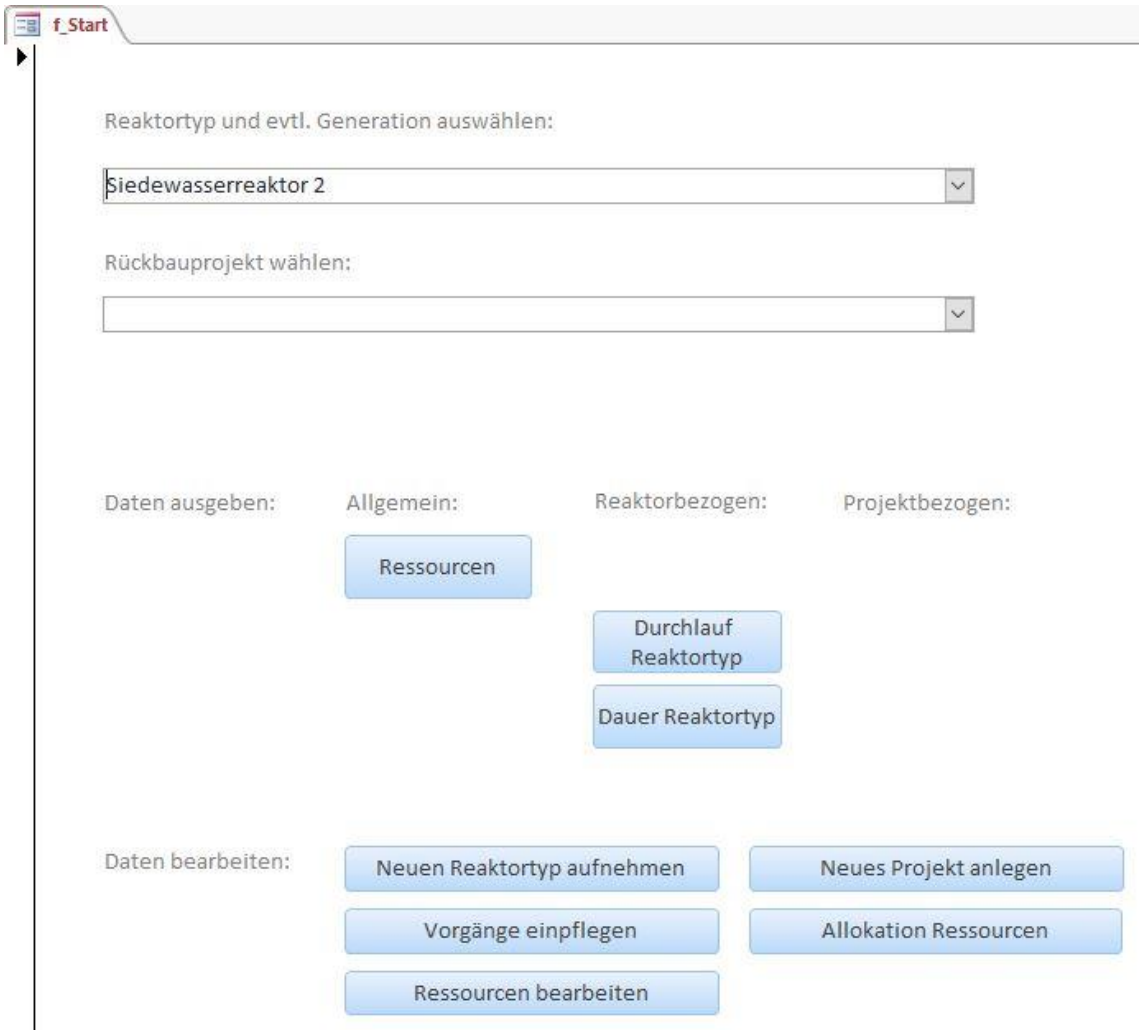


Abbildung 31: Formular ‚f_Start' mit ausgewähltem Reaktortyp

Das zweite erschienene Drop-Down-Menü bittet darum, ein konkretes Rückbauprojekt auszuwählen. Es können alle Rückbauprojekte ausgewählt werden, die den Rückbau einer kerntechnischen Anlage des im ersten Drop-Down-Menü ausgewählten Reaktortyps zur Aufgabe haben. Nachdem ein Projekt ausgewählt wurde, erscheinen weitere Schaltflächen unter der Rubrik ‚Projektbezogen' (vgl. Abbildung 32).

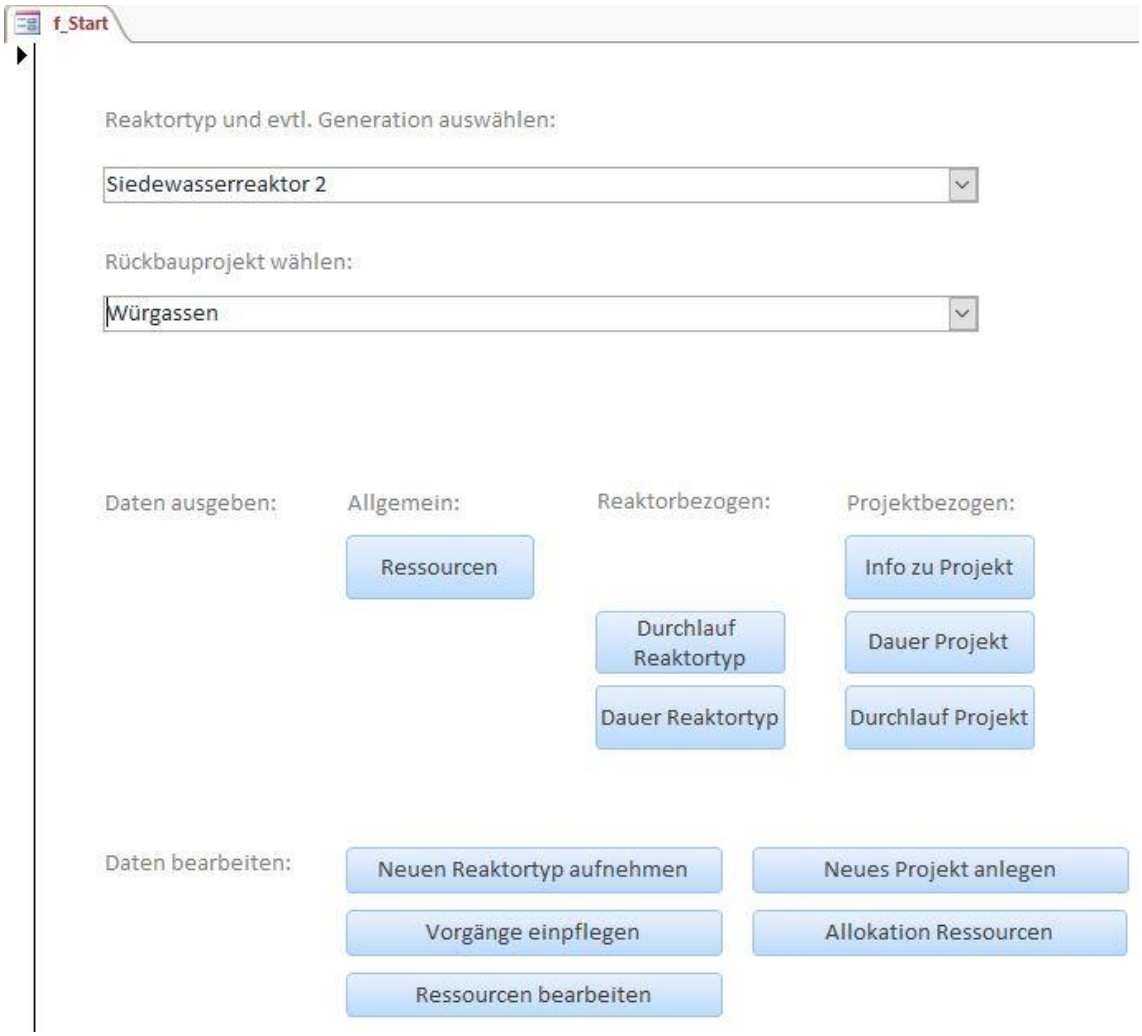


Abbildung 32: Formular ‚f_Start' mit ausgewähltem Reaktortyp und ausgewähltem Rückbauprojekt

Unter den neu erschienenen Schaltflächen finden sich zwei, die analog zu den beiden zuvor beschriebenen Schaltflächen funktionieren. Der einzige Unterschied besteht darin, dass die Daten nicht für alle Projekte ausgegeben werden, die dem ausgewählten Reaktortyp (und evtl. seiner Generation) angehörig sind, sondern nur die Daten des konkret ausgewählten Rückbauprojekts. Dies umfasst die beiden Schaltflächen mit der Aufschrift ‚Dauer Projekt' und ‚Durchlauf Projekt'.

Die Betätigung der dritten neu erschienenen Schaltfläche mit der Aufschrift ‚Info zu Projekt' gibt die dem Rückbauprojekt zugrundeliegenden Informationen aus. Dies sind neben dem Projektnamen die Restbetriebskosten, das Fälligkeitsdatum (Deadline) und der rückzubauende Reaktortyp sowie gegebenenfalls dessen Generation.

Alle durch die Schaltflächen geöffneten Berichte verfügen über die in Abbildung 33 dargestellten Schaltflächen ‚Excel Export' und ‚Zurück'. Diese befinden sich jeweils am oberen rechten Rand des Berichts. Die Schaltfläche ‚Zurück' funktioniert analog zu den in 8.1 beschriebenen Schaltflächen mit der Aufschrift ‚Zurück', indem der aktuelle

Bericht geschlossen und das Formular ‚f_Start' erneut zur aktiven Ansicht wird. Wird auf die Schaltfläche mit der Aufschrift ‚Excel Export' in einem der genannten Berichte geklickt, dann wird eine Excel-Datei mit den Abfragedaten erzeugt, die im aktuellen Verzeichnis abgespeichert und automatisch geöffnet wird.

Abbildung 33: Schaltflächen der Berichte zur Navigation oder zum Export der Daten

9 Fazit und Ausblick

In den kommenden Jahren werden viele kerntechnische Rückbauprojekte anstehen. Dabei werden eine Vielzahl an Herausforderungen an die Planung und die Durchführung solcher Projekte gestellt. Von besonderer Bedeutung sind im Zusammenhang mit dem kerntechnischen Rückbau jedoch die fehlenden Erfahrungswerte und die sich daraus ergebenen großen Unsicherheiten während der Projektausführung.

Ziel dieser Arbeit ist es, diese Unsicherheiten zu reduzieren, indem Erfahrungen im Rückbau kerntechnischer Anlagen festgehalten und für zukünftige Projekte zugänglich gemacht werden. Um dieses Ziel zu erreichen, wurde ein Expertensystem entwickelt. Hierzu wurde zunächst eine Anforderungsanalyse durchgeführt. Anschließend wurde das konzeptionelle Datenmodell entwickelt, eine Auswahl eines Datenbankmanagementsystems getroffen und das logische Design entworfen. Zuletzt wurde das Expertensystem in Microsoft Access implementiert.

Neben der Eingabe und Speicherung von Erfahrungswerten ist auch der Abruf der Daten möglich. Somit können fortlaufend Erfahrungswerte in die Datenbank eingefügt und bestehende Daten gezielt abgefragt werden. Die Abfrage der Daten umfasst sowohl die Dauern, die zeitlichen Abhängigkeiten und die Ressourcenbeanspruchung der verschiedenen Vorgänge als auch weitere Details wie beispielsweise diverse Ausführungsmöglichkeiten und die Anzahl der maximal möglichen Zyklen eines Vorgangs.

Mit dem im Rahmen dieser Studie entwickelten Expertensystem für den Rückbau kerntechnischer Anlagen wurde eine Möglichkeit aufgezeigt und umgesetzt, wie Erfahrungswerte von abgeschlossenen Rückbauprojekten für die Planung und Durchführung zukünftiger Rückbauprojekte nutzbar gemacht werden können. Allerdings weist dieser Ansatz einige Erweiterungsmöglichkeiten auf, die in zukünftigen Arbeiten angegangen werden können.

Es ist denkbar, dass in Zukunft weitere speicherwürdige Informationen der Erfahrungsdatenbank hinzugefügt werden sollen. Um dieses Ziel zu erreichen, ist es möglich die bestehenden Relationen zu verändern oder durch neue Attribute zu erweitern.

Da Microsoft Access nur für ein begrenztes Datenvolumen ausgelegt ist, lässt sich die Datenbank, wie bereits in Kapitel 6.3 beschrieben wurde, auf Microsoft SQL Server migrieren, um den gesteigerten Anforderungen gerecht zu werden.

Literaturverzeichnis

Beeger, Britta (2015): Flughafen BER – Eine Chronik des Scheiterns. In: Frankfurter Allgemeine Zeitung. Online verfügbar unter <http://www.faz.net/aktuell/wirtschaft/flughafen-ber-eine-chronik-des-scheiterns-13895339.html>, zuletzt aktualisiert am 06.03.2017, zuletzt geprüft am 28.03.2018.

Chen, Peter Pin-Shan (1976): The entity-relationship model-toward a unified view of data. In: ACM Trans. Database Syst. 1 (1), S. 9–36. DOI: 10.1145/320434.320440.

CODASYL Data Base Task Group (1970): Data base task group report to the CODASYL programming language committee. In: SIGMIS Database 2 (2), S. 11–18. DOI: 10.1145/2579329.2579336.

Codd, Edgar Frank (1970): A relational model of data for large shared data banks. In: Commun. ACM 13 (6), S. 377–387. DOI: 10.1145/362384.362685.

DB-Engines.com (Hg.) (oJ): NoSQL - DB-Engines Enzyklopädie. Online verfügbar unter <https://db-engines.com/de/article/NoSQL>, zuletzt geprüft am 05.04.2018.

DB-Engines.com (Hg.) (2018a): DB-Engines Ranking - die Rangliste der populärsten Relational DBMS. Online verfügbar unter <https://db-engines.com/de/ranking/relational+dbms>, zuletzt geprüft am 05.04.2018.

DB-Engines.com (Hg.) (2018b): DB-Engines Ranking pro Datenbankmodell Kategorie. Online verfügbar unter <https://db-engines.com/de/ranking_categories>, zuletzt geprüft am 05.04.2018.

DB-Engines.com (Hg.) (2018c): Microsoft Access vs. Microsoft SQL Server vs. MySQL Vergleich. Online verfügbar unter <https://db-engines.com/de/system/Microsoft+Access%3BMicrosoft+SQL+Server%3BMySQL>, zuletzt geprüft am 05.04.2018.

Edlich, Stefan; Friedland, Achim; Hampe, Jens; Brauer, Benjamin (2010): NoSQL. Einstieg in die Welt nichtrelationaler Web-2.0-Datenbanken. München: Hanser.

Elmasri, Ramez; Navathe, Shamkant (2011): Grundlagen von Datenbanksystemen. Bachelorausgabe. 3., aktualisierte Aufl., Bachelorausg., [Nachdr.]. München [u.a.]: Pearson Studium (Informatik).

E.ON Kernkraft GmbH (2014): Kernkraftwerk Würgassen. Erfolgreicher Rückbau. Online verfügbar unter <http://docplayer.org/35788395-Kernkraftwerk-wuergassen.html>, zuletzt geprüft am 28.03.2018.

Geisler, Frank (2014): Datenbanken: Grundlagen und Design. 5., aktualisierte und erweiterte Aufl. Heidelberg: Mitp.

Heide, Dana (2015): AKW-Rückbau: Reicht das Geld für den Atomausstieg? Online verfügbar unter <http://www.handelsblatt.com/unternehmen/energie/akw-rueckbau-reicht-das-geld-fuer-denatomausstieg/11509862-all.html>, zuletzt aktualisiert am 16.03.2015, zuletzt geprüft am 28.03.2018.

Hübner, Felix; Hünlich, Tobias; Frost, Florian; Volk, Rebekka; Schultmann, Frank (2017): Analyse des internationalen Marktes für den Rückbau kerntechnischer Anlagen: Stand und Ausblick. Online verfügbar unter <https://publikationen.bibliothek.kit.edu/1000076792>.

Hübner, Felix; Volk, Rebekka; Semme, Josua; Schultmann, Frank (2016): Improvement of nuclear decommissioning and dismantling planning via experience exchange and optimisation methods. Proceedings of the 3rd Conference on Technological Innovations in Nuclear Civil Engineering, Paris, F, 5.- 9. September 2016. Online verfügbar unter <https://publikationen.bibliothek.kit.edu/1000060408>.

Jarosch, Helmut (2016): Grundkurs Datenbankentwurf. Eine beispielorientierte Einführung für Studierende und Praktiker. 4., überarbeitete und aktualisierte Auflage. Wiesbaden: Springer Vieweg.

Lang, Stefan M.; Lockemann, Peter C. (1995): Datenbankeinsatz. Berlin, New York: Springer-Verlag.

Meier, Andreas; Kaufmann, Michael: SQL- & NoSQL-Datenbanken. 8., überarb. u. erw. Aufl. 2016 (EXamen.press).

Microsoft Corporation (oJ): Verschieben von Access-Daten in eine SQL Server-Datenbank mithilfe des Upsizing-Assistenten. Online verfügbar unter <https://support.office.com/de-de/article/Verschieben-von-Access-Daten-in-eine-SQL-Server-Datenbank-mithilfe-des-Upsizing-Assistenten-5d74c0df-c8cd-4867-8d07-e6e759d72924>, zuletzt geprüft am 05.04.2018.

Piepmeyer, Lothar (2011): Grundkurs Datenbanksysteme. Von den Konzepten bis zur Anwendungsentwicklung. München: Hanser, Carl.

Royce, Winston W. (1987): Managing the development of large software systems. In: ICSE '87 Proceedings of the 9th international conference on Software Engineering, S. 328–338.

Saake, Gunter; Sattler, Kai-Uwe; Heuer, Andreas (2013): Datenbanken. Konzepte und Sprachen. 5. Aufl. Heidelberg [u.a.]: Mitp.

Sauer, Hermann (2002): Relationale Datenbanken. Theorie und Praxis. Mit einem Beitrag zu SQL-3 von Klaus Grieger. 5., aktualisierte und erw. Aufl. München [u.a.]: Addison-Wesley.

Schlageter, Gunter; Stucky, Wolffried (1983): Datenbanksysteme. Konzepte und Modelle; mit einigen Tabellen und zahlreichen Beispielen. 2., neubearb. und erw. Aufl. Stuttgart: Teubner (Teubner-Studienbücher Informatik, 37).

Staud, Josef L. (2005): Datenmodellierung und Datenbankentwurf. Ein Vergleich aktueller Methoden. 1. Aufl. Berlin: Springer.

Südwestrundfunk (2016): Hintergrund: Chronologie der Kosten-Explosion bei S21. Online verfügbar unter <https://www.swr.de/swraktuell/bw/hintergrundchronologie-der-kosten-explosion-bei-s21/-/id=1622/did=11787024/nid=1622/im3vzl/index.html>, zuletzt aktualisiert am 15.06.2016, zuletzt geprüft am 28.03.2018.

Vanhoucke, Mario (2013): Project Management with Dynamic Scheduling. Baseline Scheduling, Risk Analysis and Project Control. 2. Auflage. Berlin, Heidelberg: Springer Berlin Heidelberg.

Working Paper Series in Production and Energy

recent issues

No. 15 Erik Merkel, Robert Kunze, Russell McKenna, Wolf Fichtner: Modellgestützte Bewertung des Kraft-Wärme-Kopplungsgesetzes 2016 anhand ausgewählter Anwendungsfälle in Wohngebäuden

No. 16 Russell McKenna, Valentin Bertsch, Kai Mainzer, Wolf Fichtner: Combining local preferences with multi-criteria decision analysis and linear optimisation to develop feasible energy concepts in small communities

No. 17 Tilman Apitzsch, Christian Klöffer, Patrick Jochem, Martin Doppelbauer, Wolf Fichtner:
Metaheuristics for online drive train efficiency optimization in electric vehicles

No. 18 Felix Hübner, Georg von Grone, Frank Schultmann: Technologien zur Zerlegung und zur Dekontamination von kerntechnischen Anlagen

No. 19 Felix Hübner, Jennifer Jana Jung, Frank Schultmann: Gefahren ionisierender Strahlung für Mensch und Umwelt in Bezug auf kerntechnische Anlagen

No. 20 Juri Lüth, Tobias Jäger, Russell McKenna, Wolf Fichtner: Photovoltaik auf Gebäuden: eine GIS-gestützte Ermittlung des Potenzials in Baden-Württemberg

No. 21 Felix Hübner, Jennifer Jana Jung, Frank Schultmann: Auswirkungen nuklearer Unfälle auf den Menschen und die Umwelt

No. 22 Felix Hübner, Uli Schellenbaum, Christian Stürck, Patrick Gerhards, Frank Schultmann: Evaluation von Schedulingproblemen für die Projektplanung von Großprojekten am Beispiel des kerntechnischen Rückbaus

No. 23 Martin Hain, Hans Schermeyer, Marliese Uhrig-Homburg, Wolf Fichtner: An Electricity Price Modeling Framework for Renewable-Dominant Markets

No. 24 Hannes Schwarz, Lars Kotthoff, Holger Hoos, Wolf Fichtner, Valentin Bertsch: Using automated algorithm configuration to improve the optimization of decentralized energy systems modeled as large-scale, two-stage stochastic programs

No. 25 Felix Hübner, Tobias Hünlich, Florian Frost, Rebekka Volk, Frank Schultmann: Analyse des internationalen Marktes für den Rückbau kerntechnischer Anlagen: Stand und Ausblick

No. 26 Jann Weinand, Russell McKenna, Wolf Fichtner: Developing a municipality typology for modelling decentralised energy systems

No. 27 Andreas Bublitz, Dogan Keles, Florian Zimmermann, Christoph Fraunholz, Wolf Fichtner: A survey on electricity market design : Insights from theory and real-world implementations of capacity remuneration mechanisms

Impressum

Karlsruher Institut für Technologie

Institut für Industriebetriebslehre und Industrielle Produktion (IIP)
Deutsch-Französisches Institut für Umweltforschung (DFIU)

Hertzstr. 16
D-76187 Karlsruhe

KIT – Die Forschungsuniversität in der Helmholtz-Gemeinschaft

Working Paper Series in Production and Energy
No. 28, April 2018

ISSN 2196-7296

www.iip.kit.edu