Hochschule für Technik und Wirtschaft Dresden (FH)

Fachbereich Informatik/Mathematik

Diplomarbeit

im Studiengang allgemeine Informatik

Thema: Konzeption und prototypische Realisierung von Schnittstellen für heterogene Zugangssysteme zu einer SQL-basierten chemischen Stoffdatenbank

eingereicht von: Steffen Leske

eingereicht am: 26.02.2010

Betreuer: Prof. Dr. oec. Gunter Gräfe (HTW Dresden)
Dr. Vinzenz Brendler (Forschungszentrum Dresden - Rossendorf)

Inhaltsverzeichnis

Verzeichnis verwendeter Abkürzungen und Glossar

ADO	**A**ctiveX **D**ata **O**bjects
ANSI	**A**merican **N**ational **S**tandards **I**nstitute
Atomare Datenbankoperationen	Datenbankoperationen, die nicht von anderen möglicherweise gleichzeitig ablaufenden Operationen beeinflusst werden können
Backend	Grafische Benutzeroberfläche (meist webbasiert) für Verwaltungs- und Administrationsaufgaben
CMS	**C**ontent **M**anagement **S**ystem
COMMIT	Befehl um Datenbankänderungen von mehreren SQL-Befehlen persistent zu machen
CONSTRAINT	Bedingungen, die beim Einfügen, Ändern und Löschen von Datensätzen in einer Datenbank erfüllt werden müssen
DBMS	**D**aten**b**ank **M**anagement **S**ystem
DCL	**D**ata **C**ontrol **L**anguage
DDL	**D**ata **D**efinition **L**anguage
DML	**D**ata **M**anipulation **L**anguage
Frontend	Grafische Benutzeroberfläche für Besucher einer Website
IFrame	HTML-Element, welches benutzt wird, um andere Webinhalte als selbständige Dokumente in einem Unterfenster des Browsers anzuzeigen
Integrität	Schutzziel in Datenbanken zur Gewährleistung von Korrektheit und Vollständigkeit
Intrusion Detection	Erkennung von Angriffen gegen ein Computersystem

JDBC	**J**ava **D**ata**b**ase **C**onnectivity
Joomla!	Ist ein populäres, volldynamisches freies Content-Management-System
Joomla!-Framework	Stellt ein Programmgerüst zur komponentenbasierten und objektorientierten Softwareentwicklung bereit und beinhaltet viele Klassen und Funktionen zur Vereinfachung der Programmierung
JSON	**J**ava**S**cript **O**bject **N**otation
MD5	**M**essage **D**igest Algorithm **5** - kryptographische Hashfunktion
Middleware	Anwendungsneutrale Programme, die so zwischen Anwendungen vermitteln, dass die eigene Komplexität und Infrastruktur verborgen bleibt
MVC-Konzept	**M**odel-**V**iew-**C**ontroller - Konzept zur Strukturierung von Programmcode in drei Einheiten (Datenmodell, Präsentation und Programmsteuerung)
OLEDB	**O**bject **L**inking and **E**mbedding **D**ata**b**ase, Datenbankschnittstelle von Microsoft
Persistenz	Fähigkeit, Daten und Objekte auf/in nichtflüchtigen Speichermedien wie Dateisystemen oder Datenbanken zu speichern
PgAdmin III	Grafische Anwendung zum Zugriff auf und zur Verwaltung von PostgreSQL-Datenbanken
Phase	In sich abgeschlossener stofflicher Bereich mit homogenen chemischen und physikalischen Eigenschaften (z.B. ein Mineral, Wasser mit allen darin gelösten Bestandteilen, oder die Gasphase)
Phasenkonstituent	Spezies, welche Bestandteil einer Phase ist

PHP	**P**HP: **H**ypertext **P**räprozessor - Skriptsprache zur Erstellung dynamischer Webseiten oder Webanwendungen
PhpPgAdmin	Webbasiertes Frontend, mit dem PostgreSQL-Datenbanken verwaltet werden können
PL/pgSQL	**P**rocedural **L**anguage/**P**ost**g**re**SQL** ist eine Programmiersprache in PostgreSQL zur Programmierung von Stored Procedures
Precompiler	Programm, welches einen Quellcode in einem Durchlauf vor dem eigentlichen Compiler bearbeitet
RDBMS	**R**elationales **D**aten**b**ank **M**anagement **S**ystem
RDO	**R**emote **D**ata **O**bjects
Relation	Bezeichnet in einer Datenbank die Abbildung auf eine Tabelle, wobei Attribute den Spaltennamen und die Attributwerte den Einträgen in den Spalten entsprechen
Rollback	Befehl zum Zurücksetzen einzelner Verarbeitungsschritte einer Transaktion
SHA-256	**S**ecure **H**ash **A**lgorithm - kryptographische Hash-Funktion mit einem 256 Bit langen Hash-Wert
Singleton-Klasse	Ist eine Klasse von der maximal nur ein Objekt erzeugt werden kann, wobei dieses üblicherweise global verfügbar ist
Spezies	Einfachste chemische Verbindung (Grundeinheit), kann ein Molekül, ein Ion, ein Gas oder ein Feststoff sein
SQL	**S**tructured **Q**uery **L**anguage
SQL-Injection	Ausnutzen einer Sicherheitslücke in Zusammenhang mit SQL-Datenbanken, die durch mangelnde Maskierung/Überprüfung von Metazeichen in Benutzereingaben entsteht

SSH	**S**ecure **Sh**ell - Netzwerkprotokoll, mit dessen Hilfe man auf sichere Art und Weise eine verschlüsselte Netzwerkverbindung mit einem entfernten Computer herstellen kann
Subselects	Unterabfragen oder geschachtelte Abfragen in SQL-Strings
TCL	**T**ransaction **C**ontrol **L**anguage
THEREDA	**The**rmodynamische **Re**ferenz**da**tenbasis
Transaktion	Eine Folge von Operationen, die als eine logische Einheit betrachtet werden
Trigger	Funktionalität von diversen DBMS zur Ereignissteuerung bei Änderungen von Daten einer Tabelle, wobei ein Programm aufgerufen wird, welches dann weitere Tätigkeiten vornehmen kann und entscheidet ob die Änderungen erlaubt oder verhindert werden
URL	**U**niform **R**essource **L**ocator - identifiziert und lokalisiert eine Ressource über das verwendete Netzwerkprotokoll und den Ort der Ressource in Computernetzwerken
Wrapper	Bezeichnet eine Komponente im Web Content Management System, mit der man externe Seiten in das CMS einbinden kann
XML	E**x**tensible **M**arkup **L**anguage - Auszeichnungssprache zur Darstellung hierarchisch strukturierter Datensätze in Form von Textdaten

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

THEREDA ist ein Projekt welches sich mit der Erstellung einer Referenzdatenbasis zu chemischen Stoffdaten für die Modellierung von Schadstofftransport befasst.

Da das Projekt ein Verbundprojekt verschiedener Forschungsinstitutionen ist und einen großen externen heterogenen Nutzerkreis hat, ist es für den gemeinsamen Datenzugriff notwendig, eine einheitliche, für alle erreichbare Plattform zum Zugriff auf die Stoffdatenbank zu schaffen. Die Datenbank selbst ist eine relationale Datenbank, da sie nach dem Datenbankmodell von Edgar F. Codd in Relationen / Tabellen abgelegt ist.

Um zu gewährleisten, dass alle Projektteilnehmer und Mitarbeiter unabhängig von ihren IT-Kenntnissen auf die Datenbank zugreifen können, erfolgt der Zugriff über eine Webapplikation. Der Internetauftritt ist bereits über das Content Management System Joomla! realisiert.

Der Zugriffsschutz ist dabei eine Anforderung mit sehr hoher Priorität, da es sich um eine Standarddatenbank für die Anwendung in Endlager- und Sanierungsprojekten handelt, was ohnehin eine sehr sensible Problematik ist. Es muss daher sichergestellt werden, dass nur berechtigte Personen Änderungen an der Datenbank vornehmen können. Dies ist besonders wichtig in Bezug auf die öffentliche Verfügbarkeit der Datenbank. Perspektivisch ist für eine garantierte Langzeitverfügbarkeit zu sorgen was auch besondere Ansprüche impliziert.

Aus diesen genannten Gründen besteht die Notwendigkeit zur Realisierung von Schnittstellen für verschiedene Zugangsmöglichkeiten zur THEREDA-Datenbank.

2. SQL-Datenbanken und deren Zugangssysteme

2.1 SQL-Datenbanken

SQL ist eine Datenbanksprache um Abfragen und Manipulationen an Daten in relationalen Datenbanken vorzunehmen. Bei relationalen Datenbanken liegt der Datenbestand in Form von zweidimensionalen Tabellen vor, welche auch als Relationen bezeichnet werden. Die Attribute der Relation werden dabei in die Spaltennamen der Tabelle abgebildet.

Es gibt vorgefertigte Datentypen für unterschiedliche Zahlentypen, Zeittypen Zeichenketten und auch Binärdaten. Die Datentypen werden für jede Tabellenspalte beim Anlegen der Tabelle festgelegt. Eine SQL-Datenbank ist daher bestens für die Anforderungen von THEREDA geeignet.

Die Sprache kann in folgende Teile untergliedert werden:

- DML (Data Manipulation Language) zum Lesen, Schreiben, Ändern und Löschen von Daten
- DDL (Data Definition Language) zum Beschreiben, Ändern oder Löschen von Datenstrukturen
- SDDL (Storage Data Description Language) zur Beschreibung der physischen Datenspeicherung und der physischen Zugriffswege zu den Daten
- DCL (Data Control Language) zur Vergabe und zum Entzug von Rechten
- TCL (Transaction Control Language) zum Bestätigen und Verwerfen von Transaktionen

Um Datenkonsistenz zu gewährleisten und verschiedene Relationen in der Datenbank zu verbinden werden Primär- und Fremdschlüssel verwendet. Ein Primärschlüssel ist eine Kombination von Spalten die in allen Tabellenzeilen unterschiedlich sind und keine fehlenden Werte enthalten. Man kann also über die Primärschlüssel jede Zeile in einer Tabelle eindeutig identifizieren.

Der Fremdschlüssel hingegen verweist auf den Primärschlüssel einer anderen Tabelle und dient somit nur zur Referenzierung eines Datensatzes. Mit diesem Konzept wird unter anderem die Datenkonsistenz zwischen Tabellen gesichert. Außerdem ist dadurch sehr einfach eine Abfrage von Daten aus mehreren Tabellen möglich.

Auch mittels Transaktionen wird in Datenbanken die Datenkonsistenz gewahrt. Es wird zum Beispiel garantiert, dass mehrere untrennbare atomare Datenbankoperationen garantiert zusammen oder auch gar nicht ausgeführt werden. Falls eine der Operationen zum Beispiel auf Grund eines Fehlers nicht durchgeführt werden konnte, müssen alle Veränderungen, welche durch die bisherigen Operationen innerhalb der Transaktion ausgeführt wurden, zurückgesetzt werden. Dieser Vorgang wird auch als Rollback bezeichnet. Um nach der erfolgreichen Ausführung aller Operationen die Veränderungen persistent in die Datenbank zu schreiben, wird ein Commit ausgeführt, das die Transaktion erfolgreich abschließt.

SQL selbst ist ursprünglich aus SEQUEL (**S**tructured **E**nglish **Que**ry **L**anguage) entstanden, welches 1974 eine von IBM definierte Abfragesprache war. Die erste Standardisierung wurde dann 1986 von ANSI bzw. 1987 von der ISO mit SQL86 (auch SQL1 genannt) verabschiedet. Diese beinhaltete vier Teile: Schemadefinitionssprache, Datenmanipulationssprache, Modul-Sprache und eingebettete Syntax. Ein großer Teil wurde aus den Spezifikationen von IBM übernommen.

In SQL89 wurden zusätzliche Features zur Integritätssicherung eingeführt, wie z.B. **DEFAULT**, **PRIMARY KEY** oder **REFERENCES**.

SQL92 (auch SQL2 genannt) wurde u. a. erweitert durch benutzerdefinierte Zeichensätze zur Internationalisierung, neue Datentypen für Datum, Uhrzeit und Zeitstempel, Tabellenoperationen wie UNION oder JOIN oder auch die Umwandlung von Datentypen mittels CAST. SQL2 wurde von der ISO verabschiedet und beinhaltet drei Versionen: ENTRY (entspricht SQL89), INTERMEDIATE und FULL SQL.

1999 wurde der Standard dann in SQL3 neu in fünf Teile strukturiert. Hauptsächliche Erweiterungen waren die Einführung von Stored Procedures, Triggern und neuen Datentypen, u. a. auch selbst definierte.

Der heute weit verbreitete Standard SQL2003 brachte Erweiterungen für XML und Funktionen zur Abbildung zwischen SQL und XML [B_VOSSE1].

Dadurch, dass SQL ein freier und genormter Standard ist, findet er auch eine sehr weite Verbreitung. Es gibt mittlerweile eine Vielzahl freier Tools, Bibliotheken und Implementierungen für die gängigen Programmiersprachen. Da die Standardisierung von SQL jedoch erst spät begonnen wurde, gibt es heute praktisch kein Datenbanksystem welches den Standard in seiner "Reinform" realisiert.

Zu den meist verwendeten und bekanntesten relationalen Datenbanksystemen, die einen Großteil des Standards unterstützen, gehören u. a.:

- Microsoft SQL Server (DBS von Microsoft, aktuelle Version 2008 [10.0] in mehreren Editionen verfügbar)
- Oracle (kostenpflichtiges DBS von Oracle, aktuelle Version 11g)
- MySQL (Open Source DBS, meist für Webanwendungen verwendet, aktuelle Version 5.5)
- DB2 (kostenpflichtiges DBS von IBM, aktuelle Version 9.7)
- PostgreSQL (Open Source DBS, aktuelle Version 8.4.2)

Der Fokus liegt in dieser Diplomarbeit aber vorranging auf PostgreSQL und teilweise auch auf MySQL, da THEREDA nur auf PostgreSQL und MySQL (für Joomla!) basiert.

2.2 Methoden zur Implementierung des externen Zugriffs auf Datenbanken

Prinzipiell werden die Zugriffsmethoden auf Datenbanken in Client-/Server-Anwendungen in folgende fünf Varianten unterschieden: Embedded SQL (ESQL), Call-Level Schnittstellen (CLI), objektorientierte Schnittstellen, skriptbasierte Schnittstellen und Java-basierte Datenbankschnittstellen. Die Hauptrelevanz gilt in dieser Arbeit jedoch den skriptbasierten Schnittstellen.

2.2.1 Embedded SQL

Embedded SQL bietet die Möglichkeit, SQL-Befehle direkt als Sprachelemente einer Programmiersprache einzubetten. Es wird häufig in C, C++, COBOL oder Pascal verwendet. Dabei werden Blöcke mit SQL-Anweisungen und Deklarationen in bestimmte Schlüsselworte eingeschlossen und dann direkt in den Quellcode einer Anwendung eingefügt. Ein großer Vorteil liegt darin, dass bei dieser Programmierweise die SQL-Syntax und Typverträglichkeit schon direkt zur Compilezeit geprüft wird. Das übernimmt entweder der Compiler selbst oder ein Precompiler wandelt die SQL-Anweisungen vorher in den Code der Hostsprache um. Es wird mit sogenannten Schnittstellenvariablen gearbeitet, die in der Hostsprache deklariert wurden und im Embedded SQL - Block einfach mit einem Doppelpunkt vorangestellt benutzt werden können (siehe Beispiel):

```
EXEC SQL BEGIN DECLARE SECTION;
      char firstName[50];
      char lastName[] = "White";
      unsigned short firstNameNull;
EXEC SQL END DECLARE SECTION;

EXEC SQL CONNECT TO server USER login.password;

EXEC SQL SELECT auFirstName INTO :firstName :firstNameNull
         FROM Authors WHERE auLastName = :lastName;

if (firstNameNull <> -1) printf("%s %s", firstName, lastName);
return (0);
```

2.2.2 Call-Level Schnittstellen (CLI)

Das Call-Level Interface ist eine Art Middleware, welche es erlaubt aus beliebigen Anwendungen auf beliebige Datenbanken zuzugreifen. Dabei werden die Befehlssätze verschiedener Datenbanksysteme auf immer die gleiche Funktionsbibliothek abgebildet. So kann der Programmierer ohne großen Aufwand unterschiedliche Datenbanksysteme ansprechen, ohne den Code für den Funktionsumfang zum Ansprechen der jeweiligen Datenbank jedes Mal neu schreiben zu müssen.

Dabei übersetzt das Call-Level Interface die Befehle des Programmes in eine Sprache die das entsprechende Datenbanksystem versteht und liefert umgekehrt auch Daten in Formaten zurück, die so vom Programm weiterverarbeitet werden können. Umgesetzt wurde das CLI-Konzept erstmals in Microsofts ODBC-Schnittstelle.

2.2.3 Objektorientierte Schnittstellen

Objektorientierte Schnittstellen wie ADO (ActiveX Data Objects), RDO (Remote Data Objects) oder OLEDB bieten Klassen an, um in objektorientierten Programmier-sprachen einfach auf Datenbanken zuzugreifen. Zum Teil werden diese verwendet um Tabellen oder Datensätze direkt im Arbeitsspeicher zu repräsentieren und direkt mit ihnen als Objekte zu arbeiten. Für die Datenanbindung gibt es wiederum weitere Klassen, die dann eine Verbindung zur Datenbank herstellen.

Zu diesen Schnittstellen gehören unter anderem auch ADO.NET, OLEDB (von Microsoft) oder auch das Document Object Model (DOM) für HTML- und XML-Dokumente. Auf diese wird aber hier im Detail nicht näher eingegangen, da sie für diese Arbeit nicht relevant sind.

2.2.4 Skriptbasierte Programmierschnittstellen

Viele Programmiersprachen arbeiten im Gegensatz zu Embedded SQL mit Programmierschnittstellen. Das heißt, es werden z.B. über bestimmte Module Funktionen bereitgestellt, welche die Kommunikation mit dem Datenbanksystem übernehmen.

Diese Kommunikation wird hauptsächlich über Zeichenketten abgehandelt, die dann beim Ergebnis der Abfrage umgewandelt werden. Syntaxfehler in SQL-Abfragen werden dadurch erst zur Laufzeit erkannt und evtl. behandelt, da das SQL-Statement nur als Zeichenkette an eine Funktion übergeben und dann erst vom Datenbanksystem interpretiert wird.

Beispiele für solche Schnittstellen sind die Perl- und PHP-Module für MySQL und PostgreSQL.

PHP

PHP ist eine modulbasierte Skriptsprache. Daher gibt es hierfür auch Module (Extensions) für verschiedene Datenbanksysteme, zum Beispiel php_pgsql oder php_mysql. In diesen Modulen werden sämtliche Funktionen zur Benutzung des Datenbanksystems bereitgestellt. Die Namen der Datenbankfunktionen sind sich in den unterschiedlichen Systemen ähnlich und beginnen immer mit einem Kürzel für das entsprechende Datenbanksystem, wie hier im Beispiel mit "**mysql_**":

```
<?php
      mysql_connect("localhost", "mysql_user", "mysql_password")
            or die("Keine Verbindung möglich: " . mysql_error());
      mysql_select_db("thereda");

      $result = mysql_query("SELECT symbol, name FROM element");

      while ($row = mysql_fetch_array($result)) {
            printf ("Element: %s  Name: %s", $row[0], $row[1]);
      }

      mysql_free_result($result);
?>
```

In **Tabelle 1** wird ein Auszug aus den wichtigsten PHP-Funktionen für PostgreSQL dargestellt.

Tabelle 1 - Die wichtigsten PHP-Funktionen für PostgreSQL

Funktionsname	**Beschreibung**
pg_connect / pg_close	Öffnet/Schließt eine PostgreSQL-Verbindung
pg_query	Gibt eine Abfrage an das Datenbanksystem weiter
pg_fetch_all pg_fetch_array pg_fetch_object pg_fetch_row	Holt alle Zeilen des Ergebnisses als Array Holt eine Zeile des Ergebnisses als Array Holt einen Datensatz als Objekt Holt einen Datensatz als numerisches Array
pg_last_error pg_last_notice	Gibt die letzte Fehlermeldung einer Verbindung zurück Gibt die letzte Notice-Meldung einer Verbindung zurück
pg_num_rows pg_num_fields	Gibt die Anzahl der Zeilen der Ergebnismenge zurück Gibt die Anzahl der Felder der Ergebnismenge zurück

2.2.5 Java-basierte Schnittstellen

In Java gibt es zum Zugriff auf die verschiedenen Datenbanksysteme eine einheitliche Schnittstelle, die Java Database Connectivity (JDBC) - Schnittstelle. Sie ist vergleichbar mit der ODBC-Schnittstelle unter Windows.

JDBC regelt den Verbindungsaufbau, leitet SQL-Anfragen an die entsprechende Datenbank weiter und stellt die Ergebnisse in einer für Java nutzbaren Form dem Programm zur Verfügung.

Für jedes Datenbanksystem sind eigene Treiber erforderlich, welche die JDBC-Spezifikationen implementieren. Diese Treiber können meist direkt von den Herstellerseiten des Datenbanksystems heruntergeladen werden. Eine Liste der aktuellen JDBC-Treiber kann auf der SUN-Website [I_SUNJD1] abgefragt werden.

2.3 Architektur und Funktionen eines DBMS

2.3.1 Architektur

Datenbankmanagementsysteme sind sehr komplexe Systeme die eine Vielzahl von Aufgaben erfüllen. Um die Architektur eines Datenbanksystems zu verstehen, ist es am Besten sich eine Datenbankabfrage im Detail anzusehen. In **Abbildung 1** wird der Weg einer Beispielabfrage schematisch dargestellt.

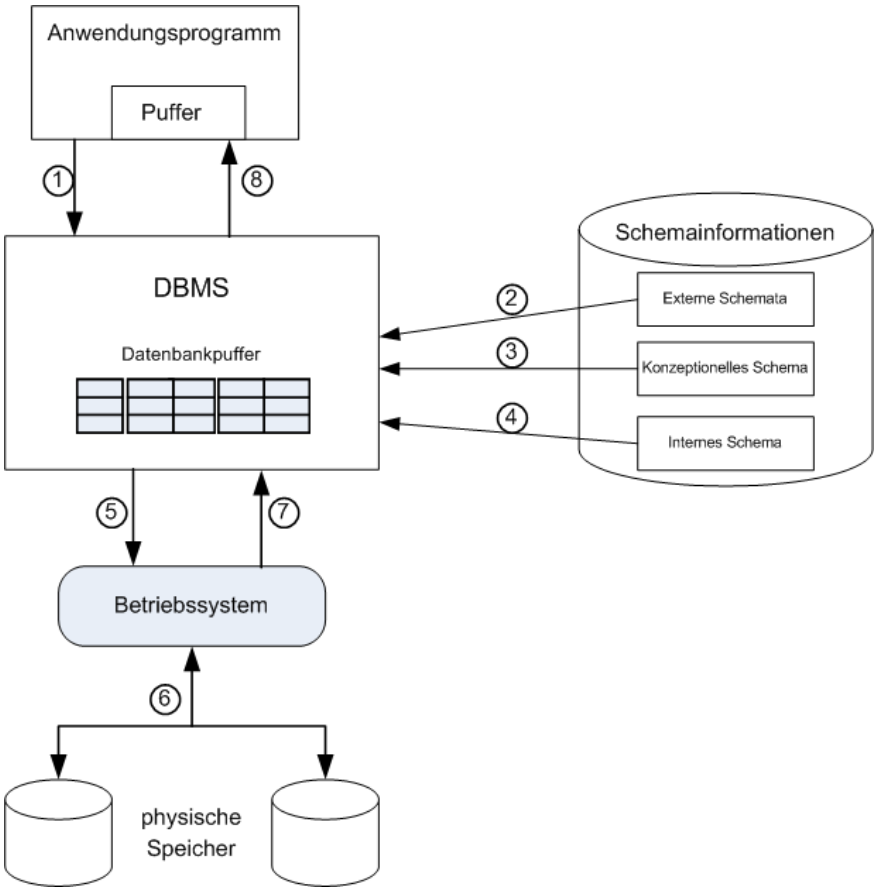


Abbildung 1 - Ablauf einer Datenbankabfrage im DBMS nach [S_GRAEF1]

Zuerst wird die Abfrage vom Anwendungsprogramm an das DBMS gesendet (1). Mit Hilfe der drei Ebenen der Schemainformationen (2-4) wird die Abfrage dann so transformiert, dass die Daten über Zugriffspfade auf der physischen Ebene abrufbar sind.

Das externe Schema (2) beinhaltet dabei die Beschreibung der Daten aus Sicht des Benutzers, also genau einen Ausschnitt aus der konzeptionellen (Gesamt-) Sicht, den der Benutzer sehen möchte.

Im konzeptionellen (logischen) Schema (3) wird die logische Gesamtheit aller Daten in der Datenbank und ihrer Beziehungen untereinander repräsentiert. Es beinhaltet also ausschließlich eine Definition des Informationsgehaltes der Datenbank insgesamt.

Das interne Schema (4) liegt der physikalischen Speicherung am nächsten, denn darin werden die verwendeten Datenstrukturen, Zugriffsmechanismen und auch die Lokalisierung der Daten auf den zur Verfügung stehenden Sekundärspeichern organisiert [B_VOSSE1].

Mit Hilfe des internen Schemas können die Daten unter Rückgriff auf vorhandene Betriebssystemfunktionen direkt aus den physischen Dateien aus dem Sekundärspeicher (6) in den Datenbankpuffer (7) geladen werden. Diese Daten müssen dann nur mit Hilfe des externen Schemas (2) wieder in die Sicht des Benutzers transformiert werden (8).

2.3.2 Funktionen

Die Funktionen eines DBMS können in fünf Gruppen unterteilt werden:

Die Hauptfunktion eines DBMS ist die **Zugriffsvermittlung**. Sie beinhaltet das Speichern und Wiederauffinden von Daten. Dazu gehören Auswahl, Anzeige, Hinzufügen, Ändern und Löschen von Daten. Diese Funktionen werden in der Data Manipulation Language (DML) beschrieben.

Zu dem kommt als weitere Funktion das **Erstellen, Ändern und Löschen von Datenbeschreibungen**, wobei diese je nach Schema in unterschiedlichen Beschreibungssprachen formuliert werden können. Die Data Definition Language (DDL) wird zur Beschreibung des externen und konzeptionellen Schemas verwendet und die Storage Data Description Language (SDDL) zur Beschreibung des internen Schemas, also der physischen Organisation der Datenspeicherung und der Zugriffswege.

Die **Integritätssicherung** dient zur Gewährleistung der Korrektheit und Vollständigkeit der gespeicherten Daten. Auf diese wichtige Funktionalität wird später noch im Detail eingegangen.

Außerdem spielt der **Zugriffsschutz** in Datenbanken auch eine sehr große Rolle. Dazu zählt beispielsweise die Sicherung des Systemzuganges mittels Nutzerkennung und Passwort (Identifizierung), die Vergabe und der Entzug von Zugriffsberechtigungen auf Datenobjekte (Autorisierung) und die Prüfung der Einhaltung definierter Befugnisse durch das DBMS. Diese Funktionen werden mit der Data Control Language (DCL) beschrieben.

Ferner bilden **Dienstprogrammfunktionen** z.B. zur Auswertung von Zugriffshäufigkeiten, Import/Export von Datenbanken oder allgemein die Unterstützung der Arbeit des Datenbankadministrators auch einen wichtigen Funktionsbereich. [S_GRAEF1]

2.4 Spezifika des PostgreSQL-RDBMS

PostgreSQL ist im Vergleich zu MySQL weitestgehend konform mit dem ANSI-SQL99-Standard und ist damit auch geeignet für komplexere Anwendungen, in denen Views, Stored Procedures und Trigger benötigt werden.

Die Rechtevergabe kann für Benutzer und Rollen (Benutzergruppen) separat geregelt werden und seit Version 8.4 können Berechtigungen auch auf Spaltenebene vergeben werden.

Weitere Spezifika sind außerdem:

- Ein umfassendes Transaktionskonzept
- Unterstützung komplexer Abfragen mit Unterabfragen (Subselects), auch geschachtelt
- Referentielle Integrität (u.a. Constraints, Fremdschlüssel)
- Views, die mit Hilfe von Regeln auch schreibfähig sein können
- Trigger und Stored Procedures in verschiedenen Sprachen (PL/pgSQL, PL/Tcl, PL/PERL, PL/Python, PL/Java, PL/PHP, ...)
- Lauffähig auf vielen Unix-Plattformen und seit Version 8.0 auch nativ unter Windows

Grenzen
(bei der aktuell für THEREDA verwendeten PostgreSQL-Version 8.3)

- Die maximale Größe einer Tabelle beträgt 32 TB.
- Die maximale Größe eines Datensatzes beträgt 1,6 TB.
- Die Datentypen VARCHAR und TEXT können nicht größer als 1 GB sein.
- Die maximale Anzahl der Spalten pro Tabelle ist abhängig von den verwendeten Datentypen und liegt zwischen 250 und 1600.

Diese Grenzen stellen für die Nutzung von THEREDA keine Einschränkung dar, da Umfang und Komplexität der Datenbank deutlich begrenzter sind.

Die aktuelle Größe der gesamten THEREDA-Datenbank beläuft sich mit Stand vom 21.02.2010 auf 1,98 MB und die maximale Spaltenzahl liegt derzeit bei 54 in der Tabelle **`interaction_variable`**.

2.5 JSON: Kommunikation über generische Zwischendateien

Die Ausgabe umfangreicher Datensets in THEREDA dient mehreren Zwecken. Zum einen kann so die Qualität der Dateneingaben intern überprüft werden. Des Weiteren können Nutzer einzelne Datensätze zu bestimmten Elementen und geochemischen Rahmenbedingungen abfragen. Schließlich können auch umfangreiche Datenbasen für die Berechnung mit geochemischen Codes erzeugt werden. Diese geochemischen Codes lassen sich in zwei verschiedene Kategorien unterteilen, innerhalb derer die jeweiligen Daten (sowohl Datentypen aus auch formale Gliederung) recht ähnlich sind. Daher ist die Erzeugung von Datenbasen in jeweils codespezifischen Formaten am effizientesten, wenn zuerst ein generisches Zwischenformat erzeugt wird und im folgenden Schritt daraus die eigentlichen Datenbasen für externe Nutzer. Die für den zweiten Schritt notwendigen Parser / Filter sind untereinander ähnlich und somit einfacher zu programmieren und zu warten.

Das generische Zwischenformat soll für den Menschen direkt lesbar sein, um auch Kontrollfunktionen erfüllen zu können. Dies impliziert nicht nur ASCII als Datei-Format, sondern auch die interne Struktur soll klar gegliedert, knapp und verständlich sein.

All diese Anforderungen werden nach Ansicht des THEREDA-Teams (und basierend auf umfangreichen Recherchen) am besten durch das JSON Format (von **J**ava**S**cript **O**bject **N**otation abgeleitet) erfüllt. Das Format ist seit vielen Jahren etabliert, in dieser Zeit gereift, mittlerweile weit verbreitet, und in einer Reihe von Werkzeugen eingesetzt

worden. Unter anderem liegen bereits Code-Beispiele für C, Java, PHP und weitere Hochsprachen vor. [I_JSON1]

Nach Entscheidung für JSON wurde ein entsprechendes Strukturkonzept für die Ablage der Daten aus THEREDA entwickelt, siehe dazu Anlage 1 - JSON - Dateistrukturdefinition. Durch die Firma Lineas wurde im Verbundprojekt die Erstellung von JSON-Dateien aus der PostgreSQL Datenbank bereits implementiert.

2.6 Interaktionen mit PostgreSQL in Joomla!

Die wesentlichen Elemente von THEREDA sind die PostgreSQL-Datenbank und das Content Management System Joomla!. Es beinhaltet mit seinem Framework bereits einige Klassen um die Arbeit mit Datenbanken zu erleichtern. Diese Klassen sind jedoch bisher nur für MySQL implementiert. Es ist aber durch Vererbung der Basis-Datenbank-Klasse (JDatabase) kein großer Aufwand diese auch für PostgreSQL zu implementieren, da die PHP-Funktionen für die Arbeit mit den Datenbanken sich in PHP sehr stark ähneln.

Um die Datenabfrage aus der PostgreSQL-Datenbank in das bestehende System einzubetten, gibt es viele Möglichkeiten. Da der Datenzugriff aber plattformunabhängig, einfach und auch für Nutzer mit geringeren IT-Kenntnissen möglich sein soll, hat sich das THEREDA-Team für die Verwendung einer Webapplikation entschieden. Dabei gibt es wiederum zwei Varianten: Entweder als externe Webapplikation, die auch unabhängig von Joomla! funktioniert oder direkt als Joomla!-Komponente im vorhandenen Content Management System auf der THEREDA-Website.

Die Vorteile einer Implementierung als Joomla!-Komponente sind unter anderem die Nutzung des Joomla!-Frameworks. Damit sind sämtliche von Joomla! gegebenen

Funktionalitäten wie Mehrsprachigkeit, direkte Einbettung in das vorhandene Layout und Zugriff auf die vorhandene Benutzer- und Rechteverwaltung möglich.

Eine externe Anwendung wäre zwar unabhängig von Joomla!, würde aber eine separate Benutzerverwaltung erfordern. Die Anwendung würde sich dann auch nur teilweise in die vorhandene Website integrieren lassen.

Es gibt noch die Möglichkeit in Joomla! eine externe Website über einen Wrapper zu integrieren. Das bringt jedoch einige Sicherheitsrisiken mit sich, z.B. wenn ein Nutzer die direkte URL der Webapplikation kennt, kann er ohne Authentifizierung die Anwendung aufrufen.

Eine Möglichkeit zur Verbesserung der Sicherheit wäre das Logging von Nutzerinteraktionen. Das bietet sich bei einer Komponente natürlich an, weil dort die ganze Zeit mit einem personalisierten Benutzer gearbeitet wird. Da bei der Einzeldatenabfrage aber nichts an den eigentlichen THEREDA-Daten verändert wird, wurde das Logging von Nutzerinteraktionen dabei als wenig relevant eingestuft.

Es wurden aber trotzdem, unter anderem zur Intrusion Detection, Maßnahmen zum Logging von Fehlern ergriffen. Näheres dazu im anschließenden Kapitel.

Weitere Sicherheitsaspekte sind in der Diplomarbeit von M. Herzog [S_HERZO1] nachzulesen.

2.7 Probleme in der Nutzung des Interface zu PostgreSQL

2.7.1 Sicherung der semantischen Integrität

Für die Prüfung der Semantik ist allein die Webapplikation verantwortlich, da diese sämtliche Nutzereingaben entgegennimmt. Das kann zum Teil mit einfachen vorgefertigten Auswahlkriterien geschehen, was jedoch bei Text-Eingabefeldern in Formularen nicht möglich ist. Hier ist es erforderlich nach dem Absenden der Formulardaten zu prüfen, ob die Eingaben gültig sind, z.B. ob wirklich Zahlen eingegeben wurden, diese in einem bestimmten Bereich liegen oder die Logik der Eingaben korrekt ist. So dass bei einem Temperaturbereich nicht evtl. ein negativer Bereich eingegeben werden kann. Außerdem darf die Formatkontrolle nicht außer Acht gelassen werden. In verschiedenen geographischen Regionen wird zum Beispiel statt dem Komma ein Punkt geschrieben. Man hat sich für das THEREDA-Vorhaben darauf geeinigt, bei allen Berechnungen die Dezimalstellen mit Punkt abzutrennen und bei Eingabe eines Kommas dieses automatisch zu ersetzen. Ein weiterer Punkt ist die Überprüfung bei Textfeldern auf bestimmte Sonderzeichen um SQL-Injections zu verhindern.

2.7.2 Signalisierung, Protokollierung und Behandlung von Fehlern

Das Auftreten von Fehlern kann nie gänzlich ausgeschlossen werden, da nicht alle Eventualitäten vorher berücksichtigt werden können. Der Zweck der Signalisierung, Protokollierung und Behandlung von Fehlern liegt hauptsächlich in der Benutzer-freundlichkeit, der Auswertung und Wartung für die Verbesserung der Software und im Aufdecken von Gefährdungen. Um eine Applikation so benutzerfreundlich wie möglich zu gestalten, sollte bei Fehleingaben in verständlicher Form darauf hingewiesen werden. Beispielsweise bei ungültigen Werten, soll ausgegeben werden welches Datum fehlerhaft eingegeben wurde und wo. Kryptische PHP- oder Daten-

bankfehlermeldungen sollten dem Nutzer vorenthalten und dafür eine einfache Meldung ausgegeben werden.

Unerwartet auftretende Fehler wie z.B. Verbindungsprobleme zur Datenbank oder falls es strukturelle Änderungen an der Datenbank gab, die noch nicht berücksichtigt worden, gilt es zu protokollieren. Je nach Grad des Fehlers soll eine verantwortliche Person per Email benachrichtigt werden. Dem Nutzer soll dabei vorrangig signalisiert werden, dass ein Fehler aufgetreten ist und bereits jemand darüber benachrichtigt wurde, ohne dabei Informationen über die Art des Fehlers oder Teile von Datenbankabfragen anzuzeigen. Bei fehlerhaften oder unsinnigen Nutzereingaben sollte zudem darauf hingewiesen und die Möglichkeit einer Korrektur angeboten werden.

Die Fehlerprotokollierung sollte in einer separaten Textdatei erfolgen, da sonst im Falle einer nicht vorhandenen Datenbankverbindung keine Protokollierung stattfinden kann. Die zu speichernden Daten sollten Datum und Uhrzeit, eingeloggter Benutzer, evtl. die IP-Adresse für die Rückverfolgung von Angriffen, der SQL-String, die Fehlermeldung der Datenbankabfrage und die aktuell besuchte Seite zur Lokalisierung des Fehlers enthalten. Um Gefährdungen aufzudecken sollte das Fehlerprotokoll regelmäßig ausgewertet bzw. an eine verantwortliche Person zugestellt werden.

3. Ist-Zustands-Analyse

3.1 Datenmodell und Architektur des THEREDA-Projektes

Entity-Relationship-Modell von THEREDA

Das in **Abbildung 2** gezeigte Entity-Relationship-Modell zeigt nur eine gekürzte Version der ganzen Datenbankstruktur. Isolierte Tabellen, Auswahllisten, temporäre Tabellen und Verwaltungstabellen wurden auf Grund der Komplexität im Datenbankdiagramm nicht dargestellt. Die Struktur des Diagrammes entspricht der THEREDA Datenbank Version 3.0.

Die eigentlichen Datentabellen von THEREDA sind für die thermodynamischen Daten die Tabellen **`data_standard_sit`**, **`data_standard_pitzer`** und **`data_variable_pitzer`** und für die Wechselwirkungskoeffizienten die Tabellen **`interaction_sit`** und **`interaction_pitzer`**. Der Unterschied zwischen den data_standard- und data_variable-Tabellen ist, dass in den data_standard-Tabellen nur feste Werte und deren Fehler für eine Temperatur von 25°C vorhanden sind und in data_variable Werte abweichend von 25°C abgelegt sind, die mit einer Temperaturfunktion über einem festgelegten Temperaturbereich gültig sind. Zu diesen Daten sind jeweils eine Temperaturfunktion, die zugehören Koeffizienten und der Gültigkeitsbereich der Funktion angegeben. Der Suffix Sit und Pitzer gibt jeweils nur an, für welches Wechselwirkungsmodell die Daten gültig sind.

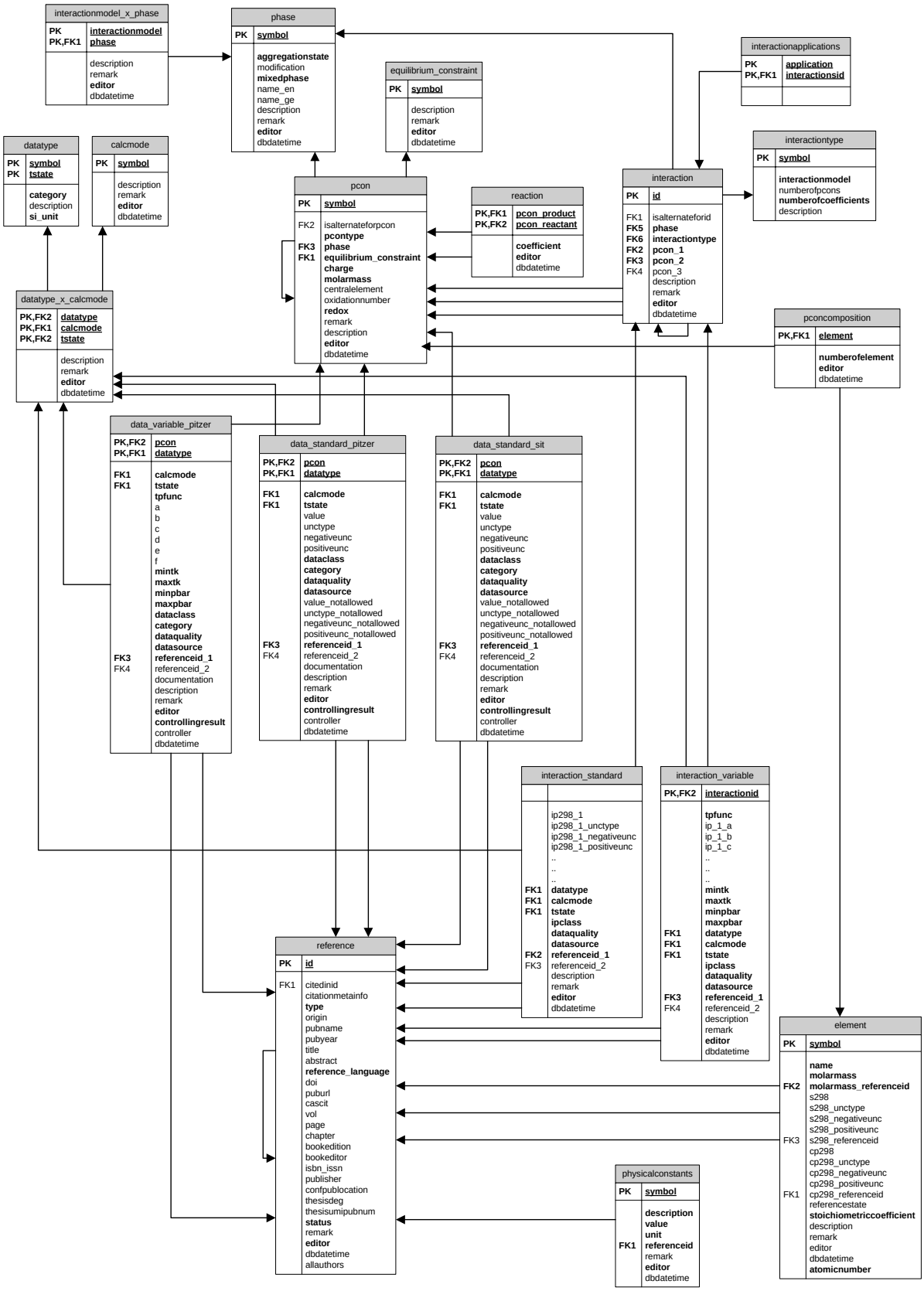


Abbildung 2 - Vereinfachtes ERM der THEREDA-Datenbank

Zu jedem Datensatz werden außerdem Qualitätsmerkmale wie **dataclass**, **category**, **datasource** und **dataquality** gespeichert, mit welchen eine Aussage über die Zuverlässigkeit, Unsicherheiten, Bewertungsprozess und Herkunft der Daten getroffen werden kann.

Um die Qualitätssicherung der Daten zu gewährleisten, ist jeder Autor eines Datums explizit auch für dessen Korrektheit verantwortlich. Dafür ist in nahezu jeder Tabelle eine Spalte **editor** hinterlegt. Um später mit der Dateneingabeoberfläche das Auditing-Konzept [S_HERZO1] umzusetzen, gibt es noch den Status in der Spalte **controllingresult**. Jedes Datum ist weiterhin eindeutig mit einem Phasenkonstituenten und einem Datentyp als Primärschlüssel beschrieben.

Phasenkonstituenten sind chemische Verbindungen, die aus mehreren Elementen zusammengesetzt sind. Die Elemente stehen mit spezifischen Informationen wie molare Masse, Wärmekapazität und Entropie in der Tabelle **element**. In **pconcomposition** werden die Phasenkonstituenten aus der Tabelle **pcon** mit den Elementen über eine Relation verbunden. So kann man später ableiten aus welchen Elementen ein Phasenkonstituent besteht oder alle Phasenkonstituenten zu einem gesuchten Element finden.

Die Wechselwirkungsdaten in den Interaction-Tabellen (**interaction_standard** und **interaction_variable**) beinhalten hauptsächlich Koeffizienten, um Wechselwirkungen zwischen mehreren Phasenkonstituenten zu beschreiben. Welche Phasenkonstituenten jeweils miteinander wechselwirken ist in der Tabelle **interaction** abgelegt. Die eigentlichen Wechselwirkungen (die sie beschreibenden Koeffizienten) sind in den Tabellen **interaction_standard** und **interaction_variable** erfasst, welche die Einträge in **interaction** referenzieren. Die Suffixe "standard" und "variable" unterscheiden in Wechselwirkungs-Daten genau für 25°C und Daten für einen variablen Temperaturbereich, wo statt den festen Werten zu jeder Spezies die Temperaturfunktion und deren Koeffizienten gespeichert sind.

Da im THEREDA-Vorhaben sehr viel Wert auf die Rückverfolgbarkeit der Daten gelegt wird, ist zu jedem Datum mindestens eine Referenz angegeben, aus der das Datum kommt. Diese Referenzen können zum Beispiel Bücher, Journale, Patente oder Reports sein. In der Tabelle **reference** sind dazugehörige Daten wie Autor, Titel, Erscheinungsjahr, bis hin zur Seitenzahl abgelegt. Es wird in den meisten Tabellen zweifach auf die Quellen referenziert, da es zum Beispiel Daten gibt, welche aus Publikationen (Sekundärreferenzen) stammen, die selbst nur auf die originale Publikation verweisen.

Die Reaktionsgleichungen für die Bildung jeder Phasenkonstituenten sind in der Tabelle **reaction** abgelegt. Die Spalte pcon_product enthält das Endprodukt (referenziert auf die Tabelle **pcon**), die Spalte pcon_reactant enthält die Ausgangsstoffe (auch in der Tabelle **pcon**) und coefficent benennt die Anzahl der jeweils pro Ausgangsstoff benötigten Moleküle in der chemischen Gleichung.

Auf die weiteren Tabellen wird hier nicht näher eingegangen, da sie weniger relevant für diese Arbeit sind.

Joomla!

Derzeit besteht THEREDA nach außen hin hauptsächlich aus dem Internetauftritt [I_THERE1]. Dieser ist durch das Content Management System Joomla! in der Version 1.5 realisiert. Die dazugehörige Datenbank ist eine MySQL-Datenbank in der Version 5.0.67. Die folgenden wichtigsten Joomla!-Komponenten und Module sind zusätzlich installiert:

- DocMan als Dokumentenverwaltung (Version 1.4.0)
  - Modul Notify für Email-Benachrichtigung bei neuen Dokumenten
  - Modul DocLink um direkte Links zur Dokumenten in Artikeln zu verwenden
  - Modul Search DocMan als Erweiterung für die Suche

- JoomFish für Mehrsprachigkeit (Version 2.0.3)
  - mit Erweiterung für DocMan
- JoomlaStats für Website-Statistiken (Version 3.0.0)
- XMap für eine Seitenübersicht (Version 1.2.2)
- Weblinks für Linklisten nach Kategorien (Version 1.5.0)
- Joomla Content Editor (JCE) als erweiterter Editor für die Inhalte (Version 1.5.7)

In Joomla! ist außerdem bereits ein Kontaktformular, eine Benutzerverwaltung und ein interner Bereich der Website integriert, der nur für registrierte Benutzer zugänglich ist.

Die THEREDA-Datenbank selbst ist eine PostgreSQL-Datenbank und ist wie in **Abbildung 3** zu sehen auf demselben Server untergebracht. Bisher existiert aber noch keinerlei Verbindung zur Website, es ist also bisher noch keine Datenabfrage oder Datenmanipulation über die Website möglich. Wie in Abbildung 3 auch zu erkennen ist, können bisher nur die Administratoren über das Webfrontend PhpPgAdmin oder über das Programm PgAdmin III Datenmanipulationen durchführen. Zur Dateneingabe bzw. Datenpflege können die THEREDA-Projektmitarbeiter bei einer Synchronisation mittels MS Excel über eine ODBC-Schnittstelle auch Daten auslesen.

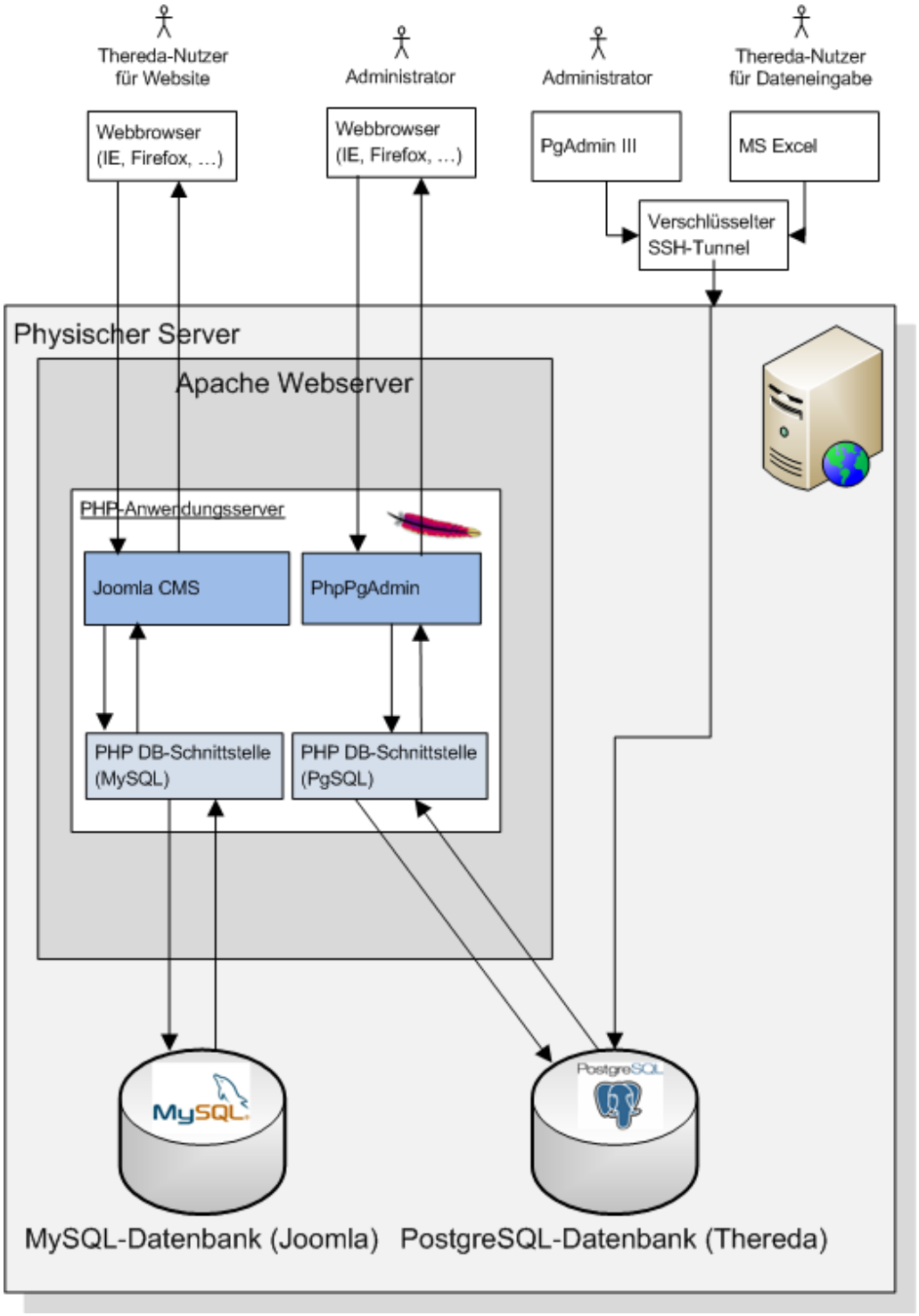


Abbildung 3 - Zugriffsschema von THEREDA im Ist-Zustand

3.2 Absicherung des Zugangs zu THEREDA

3.2.1 Sicherheitskonzept des Joomla! CMS

Der Zugang zur Website wird über das Content Management System Joomla! verwaltet. Es gibt einen öffentlichen Bereich der Website, wo allgemeine Projektinformationen, News, öffentliche Dokumente und Kontaktmöglichkeiten zugänglich sind. Diesen Bereich kann jeder Besucher der Website ohne Authentifizierung aufrufen. Es können jedoch keine Daten verändert werden.

Der nicht öffentliche Bereich der Website ist erst nach einer Registrierung zugänglich. Ein registrierter Benutzer hat Zugriff auf interne Bereiche wie zum Beispiel die Datenabfragen. Für registrierte Benutzer unterscheidet Joomla! verschiedene Benutzergruppen mit unterschiedlichen Berechtigungen. Nach der Registrierung ist ein Benutzer automatisch nur in der Gruppe "registered Users". Dieser Status erlaubt es noch nicht Veränderungen an der Website oder Datenmanipulationen vorzunehmen. Die Gruppenzuordnung kann dann manuell über den Administrator im Backend von Joomla! angepasst werden.

Ist ein Benutzer in der Gruppe Editor, so darf er Änderungen an der Website vornehmen wie zum Beispiel Texte verändern oder neue Artikel eingeben.

Zusätzlich gibt es noch eine separate Zugangsbeschränkung zum internen Bereich der Dokumentenverwaltung. Bevor der Nutzer diesen Bereich aufrufen darf, muss er ebenfalls vom Administrator im Backend bei der Dokumentenverwaltung der Gruppe THEREDA-Mitglieder zugeordnet sein. Erst dann kann der Benutzer Dokumente im internen Bereich downloaden, verändern oder neue Dokumente hochladen.

3.2.2 Zugangskonzept für die PostgreSQL-Datenbank

Der direkte Zugang zur PostgreSQL-Datenbank ist mehrfach gesichert. Der erste Zugangsschutz besteht darin, dass der Port der Datenbank, über den mittels des TCP/IP-Netzwerkprotokolls eine Verbindung aufgebaut werden kann, nach außen hin von der Firewall blockiert wird. Das heißt, man kann von außerhalb ohne Weiteres keine Verbindung zur Datenbank herstellen sondern nur direkt vom THEREDA-Server selbst aus. Für einige Benutzer wird u. a. zur Datensynchronisation eine direkte Verbindung benötigt. Dafür gibt es aber eine separate Lösung.

Es muss zuerst eine verschlüsselte SSH-Verbindung zum THEREDA-Server aufgebaut werden. Hierfür wurde auf Server-Ebene ein spezieller Nutzer eingerichtet, der nach der Authentifizierung keine weiteren System-Rechte auf dem Server hat. Über diese SSH-Verbindung kann dann ein verschlüsselter Port-Tunnel zur Datenbank aufgebaut werden in dem die Datenpakete vom Rechner des Benutzers über die bestehende Verbindung zum Server zum Port des Datenbankservers übertragen werden.

Damit kann der Benutzer eine Verbindung direkt zur Datenbank herstellen. Dies erfordert wiederum eine Authentifizierung auf PostgreSQL-Ebene. Da für diesen Zugang nur lesender Zugriff auf bestimmte Tabellen benötigt wird, ist dafür ein PostgreSQL-Nutzer mit eingeschränkten Rechten eingerichtet worden.

3.3 Bewertung von Zugangsmöglichkeiten zu PostgreSQL–Daten

Es gibt sehr viele zum Teil auch kostenpflichtige Softwarewerkzeuge zum Zugriff auf eine PostgreSQL-Datenbank, wie z.B. DbVisualizer, SQLBoss oder PGexplorer. Im Folgenden wird aber nur auf die bei THEREDA verwendeten Werkzeuge eingegangen.

3.3.1 Kommandozeilen-Tools

Für diesen Weg ist zuerst eine direkte Authentifizierung am THEREDA-Server notwendig. Sobald der Benutzer angemeldet ist, kann er über die Kommandozeile Werkzeuge wie psql, pg_dump, pg_dumpall oder pg_restore starten, welche integrale Bestandteile der PostgreSQL-Installation sind. Damit können u. a. Datenmanipulationen durchgeführt werden. Hauptsächlich werden diese Werkzeuge nur von Administratoren verwendet um strukturelle Änderungen vorzunehmen oder Datenbank-Dumps für Backups zu erzeugen oder einzuspielen.

Diese Tools sind zwar gut dokumentiert und sehr leistungsfähig, aber zur Dateneingabe und Datenpflege eher ungeeignet. Es werden sehr gute SQL-Kenntnisse vorausgesetzt und man kann Daten nur direkt mit SQL-Abfragen verändern bzw. auslesen. Es ist außerdem recht umständlich auf die Kommandozeile zu gelangen und die Anschaulichkeit der Daten ist auch nicht sehr repräsentativ, da die Ergebnismengen nur als ASCII-Text angezeigt bzw. als Export in einer Datei gespeichert werden können.

3.3.2 PgAdmin III

PgAdmin III ist ein Programm zur Verwaltung eines PostgreSQL-Datenbanksystems. Es läuft auf unterschiedlichen Betriebssystemen wie zum Beispiel Microsoft Windows, Linux und Mac OS_X. Es lassen sich alle PostgreSQL-Versionen ab 7.3 damit verwalten. Zusätzlich besitzt es Funktionalitäten zum Datenzugriff (Abfragen etc.), zur Verwaltung (Einsicht in Logfiles, Prozesse, Locks, ...) und zum Zugriff auf alle PostgreSQL-Objekte wie Trigger, Benutzer und Rollen, Views, Funktionen und Domains.

Für den Verbindungsaufbau ist der in Abschnitt 3.2.2 beschriebene Weg über den SSH-Tunnel notwendig. Alle Funktionalitäten sind natürlich nur verfügbar, wenn man an der Datenbank selbst als Benutzer mit allen erforderlichen Rechten authentifiziert ist.

PgAdmin III ist eine grafisch ansprechende und auch sehr leistungsstarke mehrsprachige Anwendung. Zur Verbindung ist jedoch derselbe umständliche Weg über den SSH-Tunnel notwendig wie auch bei den Kommandozeilen-Tools. Die grafische Oberfläche ist gut gegliedert und farblich aufbereitet. Die Anwendung umfasst sämtliche Verwaltungs- und Abfragefunktionen für das DBMS, setzt aber zur Bedienung auch genaue Kenntnis der Datenbankstruktur und ein hohes Fachwissen über Datenbanken voraus. Daher ist PgAdmin III für administrative Zwecke sehr gut geeignet, aber zur Datenpflege für normale Nutzer ungeeignet.

3.3.3 PhpPgAdmin

PhpPgAdmin ist ein in PHP geschriebenes webbasiertes Frontend zur Verwaltung von PostgreSQL-Datenbanken. Es ist direkt auf dem Webserver installiert und daher plattformunabhängig und mit einem Webbrowser über das Internet zu erreichen.

Der Zugang zu PhpPgAdmin selbst ist über eine htaccess-Datei auf dem Webserver abgesichert, die den Benutzer erst nach erfolgreicher Authentifizierung die Website aufrufen lässt. In PhpPgAdmin ist dann eine weitere Authentifizierung notwendig, welche direkt an die Datenbank weitergereicht wird.

Das Webfrontend hat den Vorteil, dass keinerlei Installation einer Software auf Client-Seite erforderlich ist, was ein großer Vorteil im Punkte Zugänglichkeit für alle Nutzer ist. Außerdem ist kein zusätzlicher Verbindungsaufbau über andere Software notwendig, sondern lediglich die Authentifizierung am Webfrontend bzw. der Datenbank selbst. Die Handhabung ist jedoch wie bei den anderen Tools nur bedingt zur Datenpflege von THEREDA geeignet, da auch hier hauptsächlich administrative Aufgaben bearbeitet werden. Die Anzeige gewünschter Daten ist nur in Tabellenform möglich, welche ohne Kenntnis der Datenbankstruktur wenig anschaulich ist.

3.3.4 Spezielle Webapplikation in Joomla!

Das Framework von Joomla! beinhaltet bereits eine Anzahl von Klassen zum Umgang mit Datenbanken. Da Joomla! bisher nur das Datenbanksystem MySQL unterstützt, ist keinerlei Funktionalität für den Zugriff auf PostgreSQL-Datenbanken darin enthalten. Im Grunde greifen diese Klassen aber auch nur auf die von PHP zur Verfügung gestellten Datenbankfunktionen zurück. Durch die Klassenstruktur und deren Möglichkeit der Vererbung ist es wie in Abschnitt 2.6 beschrieben ohne größeren Aufwand möglich eine Datenbankklasse für PostgreSQL mit Hilfe des Joomla!-Frameworks zu implementieren.

Eine Webapplikation verwendet zum Zugriff auf eine Datenbank im Allgemeinen nur einen Benutzer, der die maximal möglichen Rechte abdeckt, welche die Webapplikation benötigt. Nach diesem Grundsatz muss in der Webapplikation selbst eine Zuordnung der möglichen Funktionalitäten für bestimmte Nutzergruppen erfolgen, durch welche dann die eigentlichen Rechte zur Datenmanipulation umgesetzt werden.

Der Zugang über eine Webapplikation ist je nach Art und Umfang für mehrere Nutzergruppen geeignet. Der Verbindungsaufbau ist ohne Probleme von Nutzern mit geringen IT-Kenntnissen zu bewerkstelligen. Die Handhabung ist sehr einfach, da sich die Webapplikation um die Zusammenstellung, Aufbereitung und Ausgabe der Daten kümmert. Es sind auch keine detaillierten Kenntnisse der Datenbankstruktur erforderlich, da nur fest vorgegebene Funktionalitäten bereitgestellt werden. Die Applikation ist auch mit abschätzbarem Aufwand problemlos um weitere Funktionalitäten erweiterbar und daher für Datenabfrage und Datenpflege für ein heterogenes Nutzerspektrum gut geeignet.

3.4 Datenabfrage über Webapplikation (JAVA)

Die Firma Lineas in Braunschweig hat bereits eine Webapplikation in Java programmiert, welche sich mit der Ausgabe von Daten im JSON-Format befasst. Die Applikation besteht aus zwei Teilen. Zum einen der Teil, welcher über ein Formular die Auswahl von Elementen und einem Wechselwirkungsmodell zulässt und nach Absenden direkt eine JSON-Datei zum Download anbietet. Der andere Teil beinhaltet den Export in eine Parameter-Datei, welche direkt im Dateiformat für einen spezifizierten Code (z.B. für die Anwendung CHEMAPP) gespeichert wird.

Die Datensätze aus der Datenbank werden dabei innerhalb der Applikation über JavaBeans zur Verfügung gestellt, welche in Java oft als Datencontainer verwendet werden, um Tabellen bzw. Datensätze wie Objekte verwenden zu können.

Diese Variante der Datenabfrage ist für die Zielgruppe von THEREDA eine gute Variante, da auch hier Mehrsprachigkeit unterstützt wird und alle Funktionen über die Website selbsterklärend sind. Jedoch von der Wartung und Erweiterbarkeit ist ein erhöhter Mehraufwand notwendig, da die Applikation bei jeder Änderung erst wieder neu kompiliert/übersetzt, in ein Zip-Archiv gepackt und hochgeladen bzw. installiert werden muss, bevor die Anwendung läuft.

3.5 Dateneingabe / Datenpflege

Die Dateneingabe wird derzeit über einen eher umständlichen Weg realisiert. Es existiert ein MS Excel-Dokument, welches über einen SSH-Tunnel mit Hilfe des PostgreSQL-ODBC-Treibers eine Verbindung zur Datenbank aufbaut. Über diese Verbindung werden dann alle bisherigen Daten mit dem Excel-Dokument synchronisiert bzw. ausgelesen. Die entsprechenden Datensätze können dann in Excel editiert bzw. auch eingefügt werden. Sobald alle Eingaben abgeschlossen sind, wird das Dokument per Email an den dafür verantwortlichen Mitarbeiter gesendet. Dieser

generiert aus den Daten fertige SQL-Abfragen, welche über PhpPgAdmin eingepflegt werden und somit die Änderungen an den Daten persistent machen.

Dieser Weg ist im Moment noch sehr langwierig und umständlich und für einen normalen Benutzer ohne gesonderte IT-Kenntnisse wenig geeignet. Außerdem ist auf diese Weise das Auditing-Konzept nicht umsetzbar, welches in der Diplomarbeit von M. Herzog [S_HERZO1] näher erläutert wird. Der Zugang zum Server mittels SSH kann auch als Sicherheitsrisiko gesehen werden, wenn der SSH-Nutzer mit dem man sich am Server authentifiziert normale Benutzerrechte am Server hat. Dadurch könnten unter Umständen sensible Daten der Internetpräsenz ausgelesen werden, die sonst nicht für jedermann zugänglich sind. In der nächsten Projektphase von THEREDA soll die Dateneingabe über eine Webapplikation stattfinden, bei der auch zusätzliche Funktionalitäten und das Konzept der Qualitätssicherung umgesetzt werden.

Die Dateneingabe bzw. Datenpflege wird hier nur erwähnt, ist aber nicht Bestandteil dieser Arbeit. Allerdings soll das Gesamt-Nutzerkonzept auch die Erfordernisse der Dateneingabe / -pflege mit abdecken.

4. Anforderungsanalyse

4.1 Funktionale Anforderungen

Einzeldatenabfrage

Die Einzeldatenabfrage ist gedacht um dem Nutzer spezifische Informationen über ein gewähltes Element zu liefern. Dabei ist zu unterscheiden, ob thermodynamische Daten oder Wechselwirkungskoeffizienten abgefragt werden sollen.

Bei den thermodynamischen Daten sollen vorher spezifische Parameter wie Element, Phasentyp (gasförmig, gelöst, fest), Temperaturbereich und Wechselwirkungsmodell von Nutzer eingegeben werden. Auf der Folgeseite sollen zuerst, egal ob Datensätze gefunden wurden oder nicht, elementspezifische Daten angezeigt werden, wie z.B. Name des Elements, Ordnungszahl, molare Masse, Entropie und Wärmekapazität. Weiterhin soll eine Selektion der gefundenen Spezies erfolgen. Diese sollen einzeln oder gruppiert nach Phasentyp und Oxidationsszahl selektiert werden können. Nach dieser Auswahl sollen zu den gewählten Spezies dann die eigentlichen Datensätze angezeigt werden. Zu jedem Datensatz gehören dabei Phase, Spezies, Datentyp, Wert +/- Fehler, Maßeinheit, Datenklasse, Datenqualität, Datenquelle und bibliographische Referenz. Weiterhin sollen soweit vorhanden noch Reaktionsgleichung und zusätzliche Erklärungen zu den Daten ausgegeben werden. Ein Export zur Speicherung der Daten auf dem PC des Nutzers soll auch möglich sein. Dieser Export soll zum einen im ASCII-Format erfolgen, um maximale Flexibilität in der Weiterverarbeitung beim Nutzer zu gewährleisten, zum anderen auch im MS Excel-Format um der weiten Verbreitung dieser Daten Rechnung zu tragen.

Bei der Auswahl der Temperatur ist noch zu unterscheiden, ob Daten für 25°C oder für einen Temperaturbereich gesucht sind. Bei den Temperaturbereichen gibt es keinen

festen Wert, sondern nur eine Temperaturfunktion mit festen Parametern und eine Angabe über den Bereich der Gültigkeit dieser Funktion.

Die eigentlichen Datensätze werden dann je nach Wechselwirkungsmodell und Temperatur aus unterschiedlichen Tabellen abgefragt.

Da der Umfang der Treffer pro Abfrage sehr stark schwanken kann, sind zudem Möglichkeiten für eine einfache Einschränkung der Suche durch den Nutzer wünschenswert. Voraussetzung ist eine Vorab-Information an den Nutzer über den Gesamttrefferumfang seiner jeweiligen Suchanfrage.

Falls bei der ersten Auswahl Informationen über Wechselwirkungskoeffizienten gewählt werden, sieht die Vorgehensweise etwas anders aus: Es werden dann nur Wechselwirkungen zwischen einzelnen Phasenkonstituenten ausgegeben. Bei der Auswahl werden auch das Wechselwirkungsmodell und der Temperaturbereich berücksichtigt. Lediglich der Phasentyp soll vernachlässigt werden.

Bei der Ausgabe wird dann in einer Tabelle unterschieden in binary, lambda und ternary Datensätze, welche sich in der Anzahl der Koeffizienten und den Namen/Typen der Parameter unterscheiden. Eine Export-Funktion ist auch hier im CSV-Format und in MS Excel geplant.

Vorgefertigte Datenbasen

In diesem Bereich werden im Regelabstand von sechs Monaten gebrauchsfertige Datenbasen abhängig von dem Stand der Datenbank zum Download angeboten. Es soll unterschieden werden nach Wechselwirkungsmodell und in welchem Datenformat die Daten gespeichert sein sollen. Die Dateien liegen in einem gesonderten Verzeichnis auf dem Webserver und sollen von außerhalb nicht direkt abrufbar sein. Als Datenformate sind derzeit JSON als generisches Datenformat und weitere momentan vier Datenformate für die unterschiedlichen chemische Modellierungscodes Geochemists Workbench (GWB) [B_BETHK1], PHREEQC [Z_PARKH1], EQ3/6 [Z_WOLER1] oder CHEMAPP [Z_ERIKS1] vorgesehen.

Für diese vordefinierten Datenbasen soll eine Möglichkeit zur Verifizierung der Daten geschaffen werden, d.h. es soll im Nachhinein möglich sein zu prüfen, ob die zur Berechnung verwendeten Daten auch wirklich ein originaler Auszug aus der THEREDA-Datenbank sind und nicht nachträglich durch den Nutzer verändert wurden.

Komplexe Systeme

Für den Bereich der komplexen Systeme sollen Parameterdateien generiert werden, die einen Auszug aus der THEREDA-Datenbank über ausgewählte Elemente und ein dazu gewähltes Wechselwirkungsmodell enthalten. Sobald der Benutzer seine Parameter gewählt hat, wird diese Datei aus der Datenbank generiert und für den Nutzer zum Download angeboten. Dabei sollen die bereits unter dem Punkt „Vorgefertigte Datenbasen“ genannten Formate für die Ausgabe angeboten werden.

Die Firma Lineas hat diese Anforderung bereits in einer Java-Applikation umgesetzt. Diese muss jedoch noch in Joomla! integriert werden.

Literaturreferenzen

Eine weitere Datenabfrage soll dem Nutzer zukünftig alle Datensätze zu einer ausgewählten Literaturreferenz ausgeben. Diese sollen in tabellarischer Form angezeigt werden. Als Eingabe soll ein einfaches Suchformularfeld dienen, wo der Benutzer eine Zeichenkette als Suchterm eingeben kann.

Sinn dieser Abfragemöglichkeit ist die inverse Suche nach Daten auf Basis einer bekannten bibliographischen Referenz. Der Nutzer kann hier entweder eine Zeichenkette angeben, welche Bestandteil eines Autorennamens oder des Titels einer Publikation ist, oder er gibt direkt den mnemonischen Kurzcode an, welcher für jede Literaturstelle in THEREDA generiert wird. In einem ersten Schritt wird eine Tabelle aller dem Suchmuster entsprechenden Literaturstellen ausgegeben, welche neben den detaillierten bibliographischen Angaben pro Treffer auch Links zu den je Literaturstelle publizierten Daten enthalten. Diese Links verweisen auf:

- thermodynamische Daten bei konstant 25°C
- thermodynamische Daten mit Temperaturfunktion
- Wechselwirkungsparameter für das SIT-Modell bei konstant 25°C
- Wechselwirkungsparameter für das Pitzer-Modell bei konstant 25°C
- Wechselwirkungsparameter für das Pitzer-Modell mit Temperaturfunktion

Die dementsprechenden Datentabellen sollen sich in Gestaltung und Inhalt so weit wie möglich an die Datenausgaben im Menüpunkt "Einzeldatenabfrage" anlehnen. Die dort möglichen Einschränkungen im Temperaturbereich werden hier durch den maximal erlaubten Temperaturbereich 0 °C bis 300 °C ersetzt. Die Spalte "Reference" kann entfallen, da diese in der gesamten Tabelle die gleiche ist. Die Informationen zur bibliographischen Referenz sollen aber trotzdem oberhalb des Tabellenkopfes angezeigt werden.

Sollten für bestimmte Datenkategorien zu einer Literaturstelle keine Daten gefunden werden, ist eine entsprechende aussagekräftige Systemmeldung für den Nutzer anzuzeigen.

4.2 Nichtfunktionale Anforderungen

Rahmenbedingungen

Generell soll das Layout der bisherigen Website beibehalten werden. Die einzelnen Teilbereiche sollen soweit möglich direkt in Joomla! integriert werden. Dazu zählt auch Vorkehrungen für Mehrsprachigkeit zu treffen und die Benutzerverwaltung von Joomla! zu verwenden. Die Webapplikation soll in Kompatibilität mit den gängigsten Browsern (Mozilla Firefox 3, Microsoft Internet Explorer 6/7/8, Opera, Konqueror, Safari und Google Chrome) optimiert werden.

Für die Lauffähigkeit wird eine funktionsfähige Joomla!-Installation mit MySQL-Datenbank vorausgesetzt. Für die Funktionalität der Datenabfragen wird weiterhin ein PostgreSQL-DBMS mit einer THEREDA-Datenbank der Struktur Version 3.0 vorausgesetzt.

Die gesamte Arbeit soll im Sinne von Quelltexten und Funktionsweise gut kommentiert und dokumentiert werden.

Funktionalität

Die Software soll korrekte Abfrageergebnisse liefern und die Wertgenauigkeit der vorhandenen Daten berücksichtigen. Weiterhin ist die Integration in das vorhandene Content Management System Joomla! vorgesehen um auch eine sichere Nutzer- und Rechteverwaltung zu gewährleisten. Es dürfen zudem keine gesetzlichen Bestimmungen wie Urheberrechte oder das Bundesdatenschutzgesetz verletzt werden.

Bedienbarkeit

Die Software ist so zu konzipieren, dass jeder Nutzer mit etwas Hintergrundwissen zum Projekt diese auch ohne lange Einarbeitungszeit bedienen kann. Dazu gehören unter anderem kurze Erklärungen zu den Eingabeformularen und für ausländische Benutzer die Wahl einer anderen Sprache. Im Falle eines Fehlers sind leicht verständliche Fehlermeldungen auszugeben.

Sicherheit

Die Komponente soll von außen vor möglichen Angriffen und unberechtigtem Zugriff geschützt werden. Jeder Benutzer der auf die Daten von THEREDA zugreift muss zuvor über eine Benutzerverwaltung authentifiziert werden.

Die Sicherung der Verfügbarkeit und Integrität der Daten ist ebenfalls eine sehr wichtige Anforderung, da mit sensiblen Daten hantiert wird. Es soll keine Möglichkeit der Manipulation durch unberechtigte Nutzer gegeben werden.

Die Datensicherung geschieht bereits automatisiert auf Server-Ebene in dreifacher Ausführung auf zwei unterschiedlichen physisch und räumlich getrennten Servern. Eine detaillierte Erläuterung dazu ist im Dokument THEREDA Backup Strategien [S_LESKE1] nachzulesen.

Zuverlässigkeit

Es soll auch bei fehlerhaften Eingaben ein definierter stabiler Zustand beibehalten werden. Im Falle des Fehlers (z.B. Browserabsturz) soll es ohne großen Aufwand möglich sein die Daten wieder zu gewinnen.

Erweiterbarkeit

Die Software soll gut strukturiert sein um spätere Erweiterungen oder Änderungen ohne übermäßige Einarbeitung in Quellcodes durchführen zu können. Die Wartung der Software oder Anpassung an andere Umgebungen, z.B. eine Testumgebung, sollte ohne weiteres durchführbar sein. Außerdem sollen Änderungen an Teilen der Software ohne Einfluss auf andere Teile durchgeführt werden können.

Performance

Es sollen keine unnötigen Rechenoperationen oder Datenbankabfragen ausgeführt werden, um selbst bei hohem Nutzeraufkommen möglichst schnelle Antwortzeiten zu gewährleisten. Die Zuverlässigkeit und Sicherheit der Software soll dadurch auch in keiner Weise beeinflusst werden. Der Bedarf an Speicherplatz zur Datenspeicherung sollte die Grenzen des aktuellen Webservers (ca. 500 GB) auch langfristig nicht überschreiten. Ebenfalls soll auch bei komplexeren Anfragen das durch PHP festgelegte Arbeitsspeicherlimit von 128 MB pro Anwendungsausführung bzw. Webseitenaufruf nicht überschritten werden.

5. Konzept

5.1 Architektur des Systems

THEREDA ist derzeit auf einem Server mit physisch eigenständiger Hardware bei einem externen Provider untergebracht. Das Hosting selbst liegt jetzt bei der Firma Datacenter Germany, in deren Räumen auch der THEREDA Server steht. Es ist ein AMD Phenom 9650 Quad-Core mit 4x1,2 GHz, 4 GB Arbeitsspeicher und 500 GB Festplattenkapazität. Demnach ist der Server also bestens für Stabilität und hohe Nutzeraufkommen in der Zukunft ausgerüstet. Wie auf dem Architekturschema in **Abbildung 4** zu sehen ist, läuft auf dem Server das Betriebssystem Suse Linux 11.1 und es ist ein Apache Webserver Version 2.2.10 mit PHP 5.2.11 und Tomcat Application Server (für Java) installiert. Außerdem sind noch die Datenbanksysteme MySQL in Version 5.0.67 (für Joomla!) und PostgreSQL in Version 8.3.8 für die eigentliche THEREDA-Datenbank eingerichtet [Z_THERE1].

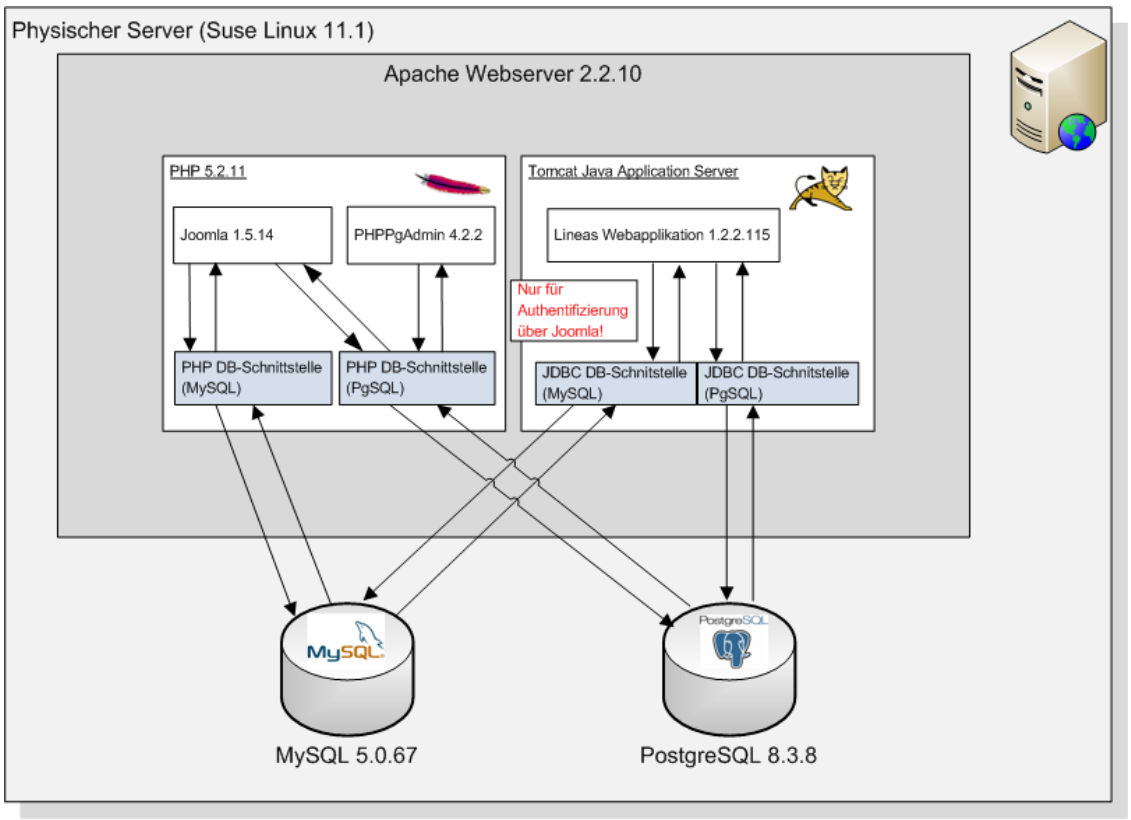


Abbildung 4 - Architekturschema des THEREDA-Servers

5.2 Schnittstellen für Datenabfrage

5.2.1 Java-Webapplikation mit Authentifizierung über Joomla!

Java läuft auf dem Webserver als eigenständige Applikation, die über eine separate URL aufgerufen wird. Daher lässt sich diese auch nicht ohne weiteres in ein PHP-basiertes Content Management System integrieren. Man kann die Java-Applikation aber zum Beispiel als ein unsichtbares Fenster (IFrame) in eine vorhandene Website einbinden. Die Applikation wird dann völlig unabhängig in diesem Fenster geladen und angezeigt. Diese Unabhängigkeit bringt jedoch den Nachteil mit sich, dass keine direkte Integration der vorhandenen Nutzerverwaltung möglich ist. Die Problematik die sich dabei ergibt: Wie kann man einen in Joomla! angemeldeten Benutzer in der Java-Anwendung authentifizieren? Die Lösung des Problems ist nicht ganz so trivial: Joomla! authentifiziert seine Benutzer bei der Anmeldung über die User-Tabelle in der MySQL-Datenbank. Bei jeder erfolgreichen Anmeldung wird zusätzlich eine Session in der Datenbank angelegt um bei einem weiteren Seitenaufruf nachzuweisen, dass der Nutzer sich bereits erfolgreich authentifiziert hat. Zu dieser Session gehören zwei Teile: Einmal der Eintrag in der MySQL-Datenbank, wo Informationen wie Login-Name, Uhrzeit, Session-ID und noch einige andere Informationen stehen. Und zum anderen der Teil, welcher in Form eines Cookies auf Nutzerseite gespeichert und bei jedem Seitenaufruf mit übertragen wird. In diesem Cookie steht meist die Session-ID, die beim Login in der Datenbank angelegt wurde (siehe Abschnitt 2 in **Abb. 5**).

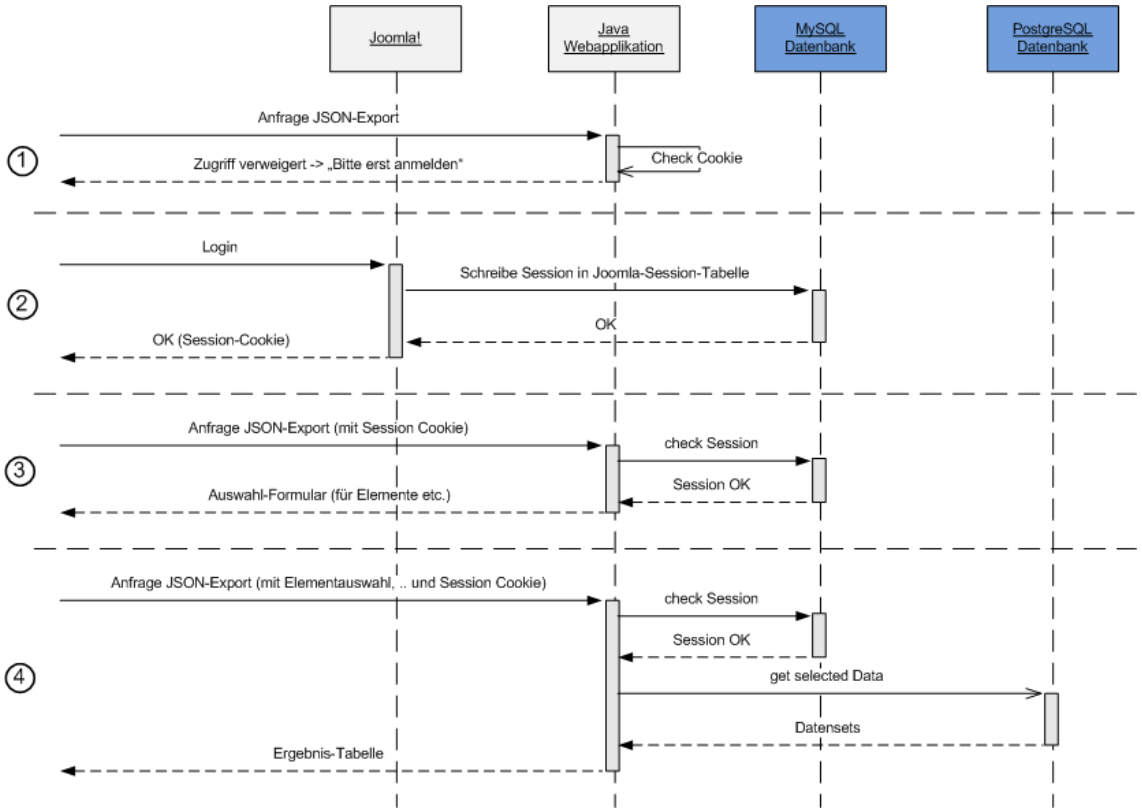


Abbildung 5 - Sequenzdiagramm: Java Webapplikation mit Auth. über Joomla!

Dieser Cookie wird natürlich beim Aufruf der Java-Webapplikation auch mit übertragen und kann auf Java-Ebene auch ausgelesen werden. Über diesen Weg kann in der Java-Applikation geprüft werden, ob ein Nutzer bereits eingeloggt ist und welcher Nutzergruppe er angehört. Es ist grob gesagt nur notwendig die Session-ID aus dem Cookie mit der Session in der MySQL-Datenbank von Joomla! zu verifizieren (siehe Abschnitt 3 in Abb. 5). Danach kann dem Nutzer der Zugriff auf die gewünschte Funktionalität gewährt oder mit einem Hinweis verweigert werden (siehe im Abschnitt 1 bzw. 4 in Abb. 5).

5.2.2 Spezifika der Schnittstelle in Joomla!

Das Joomla!-Framework bietet bereits von Haus aus eine Vielzahl von Klassen um die Entwicklung von Webanwendungen zu vereinfachen. Für den Zugriff auf Datenbanken gibt es das Package **`database`**. Es beinhaltet unter anderem die abstrakte Basisklasse JDatabase, welche alle Methoden zur Arbeit der Datenbank bereitstellt, wie z.B. Verbindungsmanagement, Datenabfrage und Rückgabe der Datensätze in verschiedenen Formaten oder Fehlerabfrage. Um Joomla! möglichst unabhängig von einem bestimmten Datenbanksystem zu machen, hat man die Implementierung der Methoden zum Datenbankzugriff in eine abgeleitete Klasse ausgelagert. So gibt es im Framework bereits die Klassen **`JDatabaseMySQL`** und **`JDatabaseMySQLi`**, die Funktionen zum Zugriff auf MySQL-Datenbanken implementieren (siehe **Abb. 6**). Für PostgreSQL ist eine solche Klasse noch nicht vorhanden, lässt sich aber sicher mit überschaubarem Aufwand implementieren, da sich die Befehle zum Zugriff auf PostgreSQL-Datenbanken in PHP nur gering zu denen von MySQL unterscheiden.

Außerdem gibt es eine abstrakte Basisklasse **`JTable`**, mit der es möglich ist Tabellen und Datensätze wie Objekte zu behandeln und so in der Webapplikation vorzuhalten. Das ist in der Hinsicht sehr sinnvoll, da viele vom Joomla!-Framework bereitgestellte Funktionen zur Manipulation von Daten über Webformulare diese Klasse benutzen. Zur Benutzung muss eine von **`JTable`** abgeleitete Klasse implementiert werden, welche als Attribute alle Spaltennamen der originalen Tabelle bekommt. Im Beispiel aus Abbildung 6 ist das die Klasse **`JTableConfig`**.

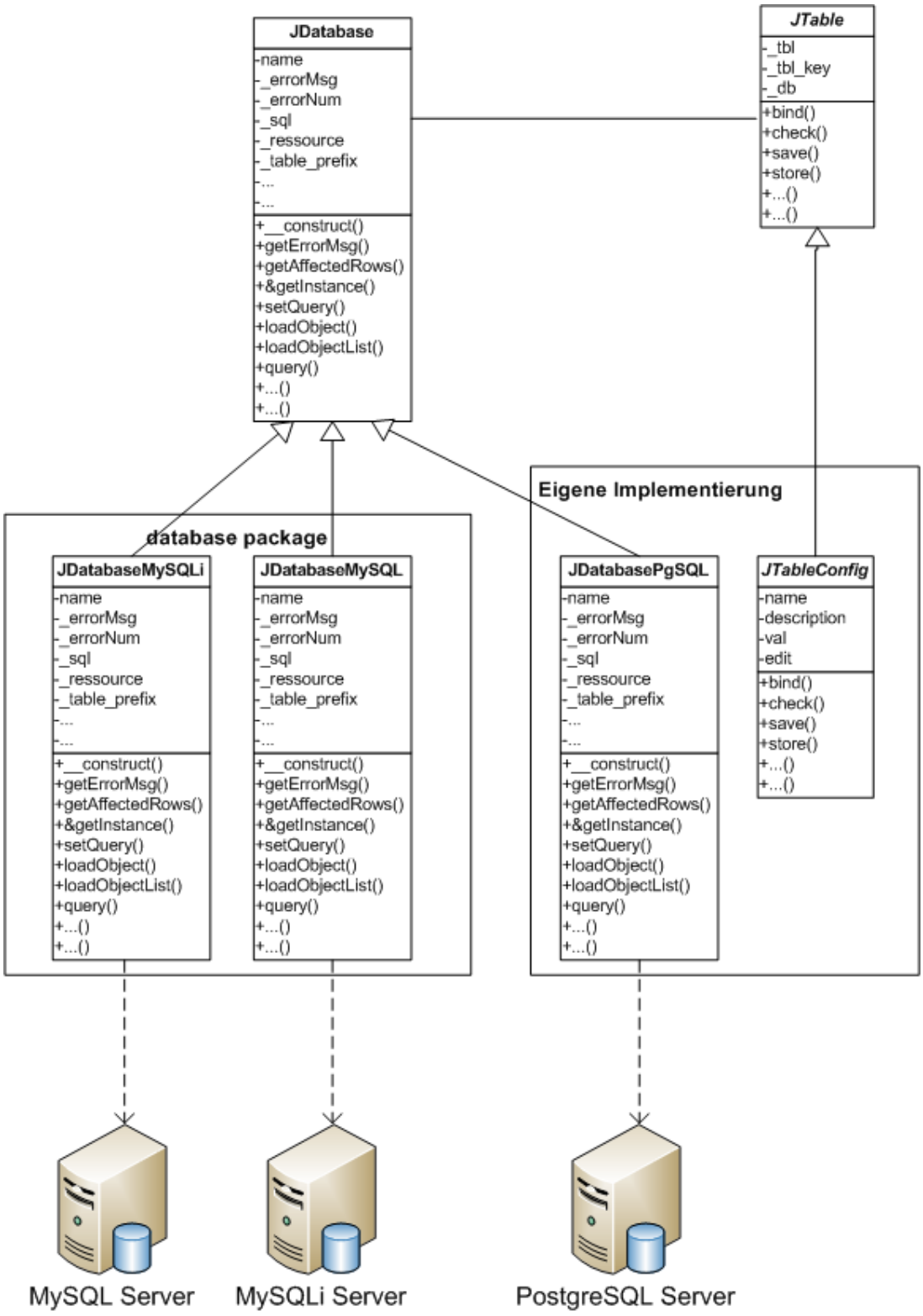


Abbildung 6 - Klassendiagramm für Datenbankzugriff in Joomla!

5.2.3 "Heterogenität" durch verschiedene Nutzergruppen in unterschiedlichen Anwendungen

THEREDA ist ein Verbundprojekt mit einer Vielzahl von Instituten, welche jeweils mit unterschiedlichen Anwendungen zur Berechnung chemischer Prozesse arbeiten. Das beruht zum Teil auf unterschiedlichen Einsatzgebieten, ökonomischen Zwängen, Firmenpolitik bzw. ist es mit der Zeit so gewachsen. Um nun zu gewährleisten, dass alle Nutzer in Ihren Anwendungen mit den Daten von THEREDA arbeiten können ist es notwendig, diese in verschiedenen Formaten auszugeben. Grundsätzlich werden von den Anwendungen dieselben Inhalte verwendet, jedoch ist der Dateiaufbau unterschiedlich und für einige Formate müssen noch Hilfsdaten wie z.B. Stützstellen für Funktionen berechnet werden.

Weitere Heterogenität herrscht auch bei den verwendeten Tools und bei den Fähigkeiten der Projektmitarbeiter, da viele hauptsächlich auf das entsprechende chemische Fachgebiet konzentriert sind und andere wiederum Verwaltungs-, IT- und Datenbankaufgaben im Projekt koordinieren. Somit kommen für die Nutzer durch unterschiedliche Arbeitsaufgaben auch verschiedene Zugangssysteme zum Einsatz, was im nächsten Unterkapitel noch näher erläutert wird.

Die Langzeitaspekte sind dabei auch zu berücksichtigen, da die ersten Konzepte für THEREDA bereits vor acht Jahren entwickelt wurden und THEREDA in diesem Prozess auf vielen Gebieten gewachsen und erweitert worden ist. Die jetzigen Konzepte werden sicher auch über die nächsten 40 Jahre noch im Einsatz bleiben. Künftige Entwicklungen sind dabei noch nicht vorauszusehen. Dadurch kann in Zukunft auch weitere Heterogenität in den Nutzermodellen, Zugriffsmodellen oder anderen Bereichen entstehen.

5.3 Benutzermodelle

5.3.1 Benutzerklassen

Es gibt bei THEREDA vier Klassen von Benutzern für die teilweise auch unterschiedliche Zugangsmodelle in Frage kommen:

- Besucher
- Interessierte Nutzer
- THEREDA-Mitglieder
- Administratoren

Besucher

Als Besucher werden alle Nutzer bezeichnet, welche lediglich das Projekt kennenlernen und sich darüber informieren wollen. Sie sind keine registrierten Benutzer, haben somit auch nur Zugriff auf öffentliche Projektinformationen. Datenabfragen sind für diese Benutzergruppe demnach auch nicht möglich. Da diese Nutzerklasse nur eher als normaler Fernnutzer eingestuft wird, besteht die einzige Zugangsmöglichkeit zu THEREDA über den Internetauftritt mit einem Webbrowser.

Interessierte Nutzer

Zu den interessierten Benutzern zählen diejenigen, welche sich für das Projekt interessieren, evtl. selbst mit den Daten von THEREDA arbeiten wollen und bereit sind, sich dafür zu registrieren. Es sind also registrierte, personalisierte Nutzer mit einem eindeutigen Login, welche auf die Daten zugreifen. Die Klasse dieser Personen sind meist wissenschaftliche Mitarbeiter von diversen Institutionen oder wissenschaftlichen

Einrichtungen mit sehr gutem chemischen Hintergrundwissen, aber mit eher geringen IT-Kenntnissen. Um den Zugang für diese Benutzer möglichst einfach zu gestalten ist auch hier nur der Zugang über die Internetpräsenz mit einem Webbrowser möglich.

THEREDA-Mitglieder

THEREDA-Mitglieder sind Projektmitarbeiter welche für die Dateneingabe und Datenpflege verantwortlich sind bzw. auch organisatorische/inhaltliche Veränderungen an der Website durchführen dürfen. Dieser Personenkreis hat tiefgehendes Wissen in Bezug auf das Projekt und kann als Experte in chemischem Sachverständnis gesehen werden. Für diese Gruppe gibt es zum Einen den Zugang über den Internetauftritt, wo auch erweitere Zugriffsmöglichkeiten zum Verändern des Inhaltes und Zugriff auf interne Dokumente möglich ist. Des Weiteren können diese Personen nicht nur über die Website Daten abfragen, sondern haben auch direkten Zugriff auf die Datenbank (momentan über die Synchronisation mit einem MS Excel Dokument) zur Dateneingabe und Datenpflege. Die Dateneingabe soll zur Vereinfachung später nicht mehr über MS Excel vorgenommen werden, sondern auch über eine Webapplikation um den Benutzern mit geringen IT-Kenntnissen den Prozess zu erleichtern. Zudem kann dadurch mehr Funktionalität und Qualitätssicherung geboten werden.

Administratoren

Zu den Administratoren zählt nur eine kleine Gruppe von Personen, welche ausgeprägtes Expertenwissen auf dem Gebiet der IT bzw. auch Chemie haben. Da hier sehr unterschiedliche Verwaltungsaufgaben anfallen, gibt es mehrere Zugangs-möglichkeiten. Die einfachen Funktionen sind wie bei den THEREDA-Mitgliedern über den Internetauftritt zugänglich. Dazu gehören Datenabfragen, Verwaltung interner und öffentliche Dokumente und das Verändern von Inhalten in Joomla!. Weiterhin ist auch

noch ein Zugang zum Backend von Joomla! möglich, um neuen registrierten Benutzern erweiterte Berechtigungen zuzuweisen oder auch strukturelle Änderungen an der Website vorzunehmen. Außerdem haben die Administratoren Zugang zu den Tools für die Datenbank-Administration um gegebenenfalls strukturelle Änderungen oder andere Operationen an der Datenbank vornehmen zu können. Dies betrifft nicht nur die Datenbank sondern auch den Webserver, welcher für Verwaltungsaufgaben über einen separaten SSH-Zugang erreichbar ist. Das setzt für die Klasse der Administratoren die höchste Verantwortlichkeit und für die unterschiedlichen Zugangsvarianten auch sehr gute IT-Kenntnisse voraus.

5.3.2 Zugangsmodelle und Authentifizierung

Es gibt bei THEREDA für die heterogenen Nutzerklassen auch unterschiedliche Zugangsmodelle, die jeweils verschiedene Funktionalitäten bereitstellen (siehe **Abb. 7**).

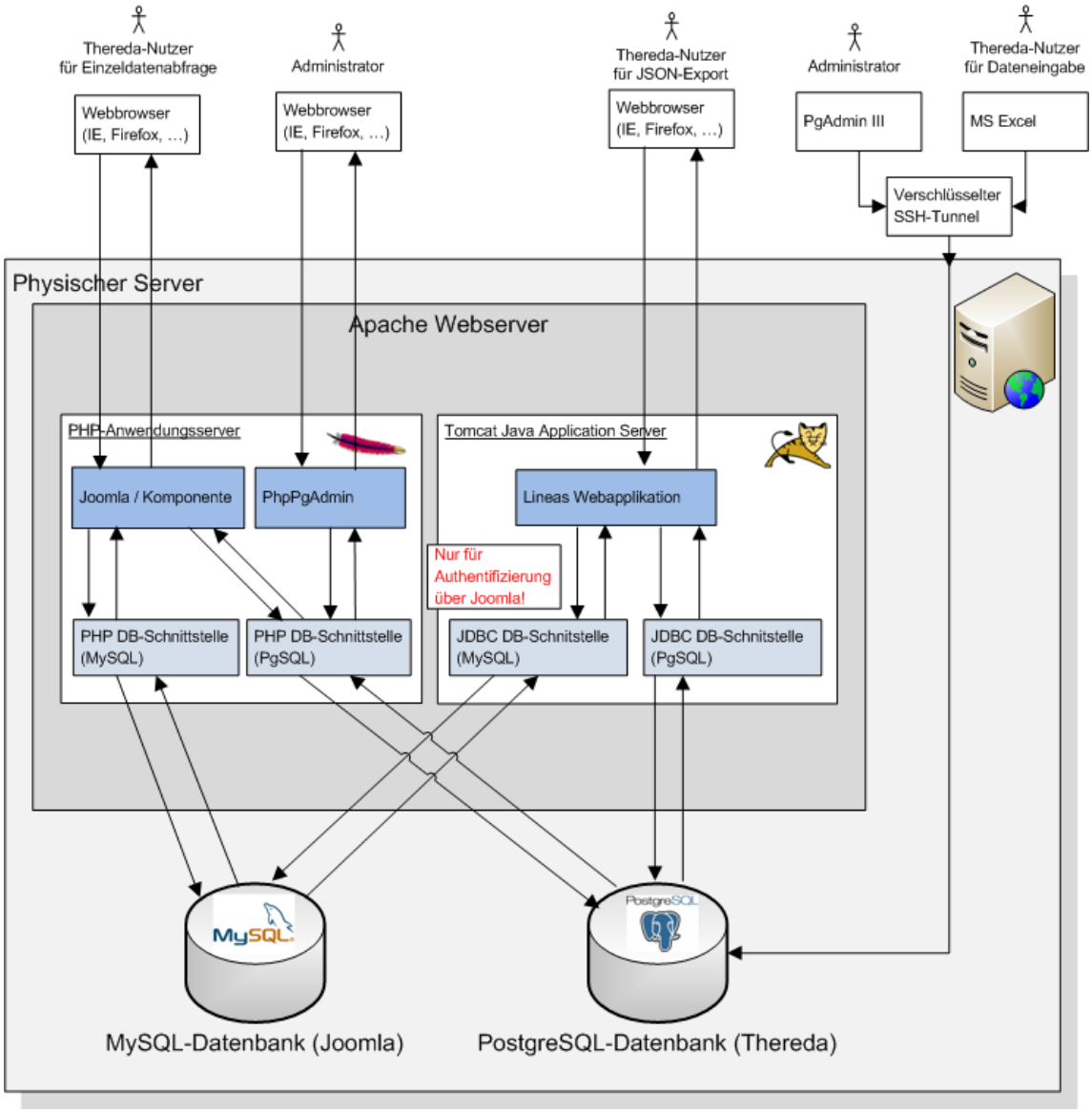


Abbildung 7 - Zugriffsschema von THEREDA mit neuen Webapplikationen

Es werden bei THEREDA derzeit drei Zugangsmodelle unterschieden. Für die Datenbank-Administration gibt es einen Zugang mittels PhpPgAdmin über die Website

und einen Zugang mit einer Clientsoftware wie PgAdmin III um den Administratoren Änderungen und Verwaltungsoperationen an der Datenbank zu ermöglichen. Die eigentliche Authentifizierung und Rechtekontrolle wird dabei hauptsächlich über das PostgreSQL-DBMS geregelt.

Weiterhin gibt es noch den Zugang direkt zum Linux-Server, welcher eine eigene Nutzerverwaltung hat. Dieser Zugang ist nur für die Administration des THEREDA-Servers gedacht, wird aber derzeit über ein separates Nutzerkonto auch für die Dateneingabe und -pflege von den THEREDA-Mitgliedern verwendet. In Zukunft soll dies aber vorrangig über die Weboberfläche geschehen.

Der Hauptzugangspunkt ist bei THEREDA der Zugang über Joomla! als Content Management System. Er findet Verwendung für alle Nutzerklassen, da die Nutzerverwaltung von Joomla! auch für den Datenexport (in JSON und andere generische Codes) verwendet wird und in Zukunft auch für die Dateneingabe und -pflege verwendet werden soll. Dafür wurde von M. Herzog bereits ein Konzept für Nutzergruppen entworfen [S_HERZO1]. Dieses im Folgenden etwas abgeänderte Schema (**Abb. 8**) soll in Zukunft alle mit THEREDA arbeitenden Nutzergruppen abdecken.

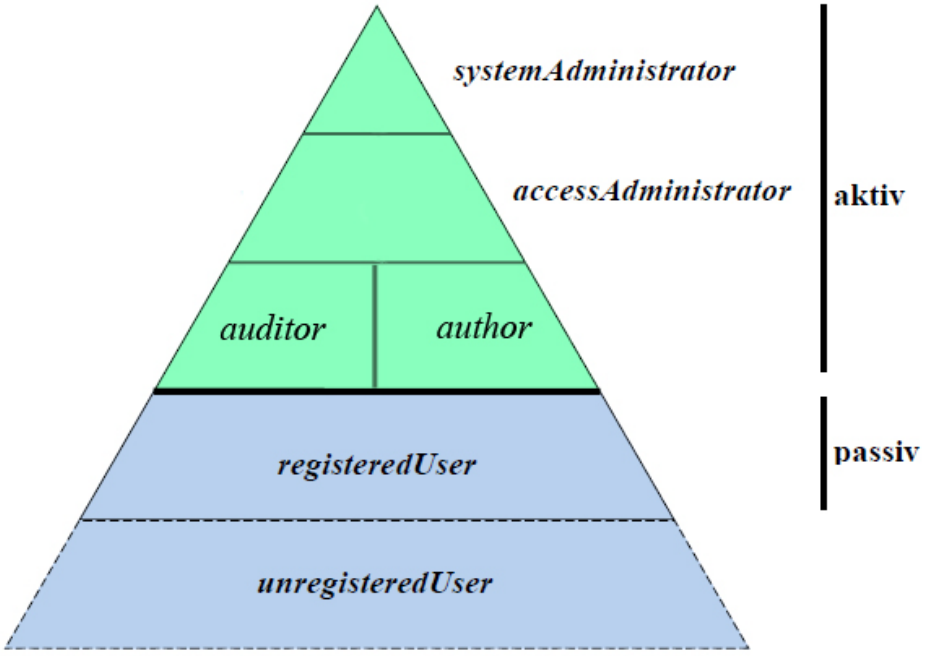


Abbildung 8 - Nutzergruppendiagramm nach [S_HERZO1]

Die **unregistered User** haben als Websitebesucher nur die Berechtigung öffentliche Daten einzusehen und sich über das Projekt zu informieren. Die **registered User** hingegen sind bereits authentifizierte und personalisierte Nutzer und können auch auf die Datenabfragen zugreifen. Beide Gruppen sind jedoch passive Benutzer, da sie keinerlei Möglichkeit haben Datenmanipulationen vorzunehmen. **Autoren** und **Auditoren** sind Nutzer, welche Daten an der Website verändern dürfen und, wenn die Oberfläche dafür fertiggestellt ist, auch Dateneingabe und -pflege vornehmen. Dabei ist zu beachten, dass die Autoren nur temporär zur Qualitätssicherung den Status Auditor erhalten. Der **Access-Administrator** übernimmt die Verwaltung der Nutzer und Zuordnung in die Nutzergruppen und der **System-Administrator** ist für Wartungsaufgaben und strukturelle Änderungen verantwortlich.

Bei all diesen Gruppen wird in den Webapplikationen ein Mapping auf die vorhandenen Joomla!-Gruppen vorgenommen, d.h. die hier angegebenen Gruppen werden den vordefinierten Joomla!-Gruppen im CMS zugeordnet.

5.4 Identitätsmanagement

5.4.1 Abspeicherung von Nutzerprofilen und -daten

Für die Arbeit an der THEREDA-Datenbank ist es erforderlich, bestimmte Interaktionen mit der Datenbank einem personalisierten Benutzer zuzuordnen. Joomla! bietet dafür bereits eine fertige Nutzerverwaltung, die verwendet werden kann um einen Nutzer für den Zugriff zur Datenbank zu authentifizieren. Um zum Beispiel selbst zusammengestellte Abfragen zu speichern oder aufzurufen werden Abfragedaten wie Temperatur, gewählte Elemente, Wechselwirkungsmodell, Phasentyp und ausgewählte Spezies unter einer ID in der Joomla!-Datenbank gespeichert und dem Nutzer über die user_id zugeordnet.

Ein Nutzer kann somit nur auf die eigenen Nutzerdaten sowie die eigene vorgehaltene Einzeldatenabfrage zugreifen. Sämtliche nutzerbezogenen Daten werden nur so lange gespeichert wie der Nutzer aktiv bei THEREDA registriert ist.

5.4.2 Nutzergruppen und Berechtigungen in Joomla!

In Joomla! gibt es bereits sechs vordefinierte Benutzergruppen. Für das Frontend stehen hier Autor, Editor und Publisher zur Verfügung und im Backend sind Manager, Administrator und Super Administrator definiert.

Ein Mitglied der Gruppe Autor kann im Frontend Inhalte ins System eingeben und eigene Inhalte bearbeiten. Die eingegebenen Inhalte müssen jedoch von einem Mitglied der Backend-Usergruppen zur Veröffentlichung freigeschaltet werden.

Als Erweiterung zum Autor kann ein Mitglied der Gruppe Editor jegliche Inhalte im Frontend editieren. Auch die von ihm eingegebenen Inhalte müssen erst zur Veröffentlichung freigeschaltet werden.

Als Erweiterung zum Editor kann ein Mitglied der Gruppe Publisher Inhalte direkt veröffentlichen, so dass diese nicht erst vom Administrator zur Veröffentlichung freigeschaltet werden müssen.

Ein Mitglied der Gruppe Manager, die als erste Benutzergruppe Zugriff auf das Backend hat, kann dort das Menü bearbeiten. Außerdem kann ein Manager – analog zum Frontend - Inhalte einfügen sowie Items aller Nutzergruppen bearbeiten und publizieren. Zusätzlich kann er im Backend Bereiche oder Kategorien bearbeiten und erstellen. Er hat Zugriff auf Statistiken und den Media Manager.

Ein Mitglied der Gruppe Administrator hat als Erweiterung zum Manager Zugriff auf die Verwaltung von Komponenten, Modulen und Plugins und kann auch neue installieren.

Zudem hat er Zugriff auf die Benutzerverwaltung und kann dort Nutzungsrechte bis hin zum Administrator vergeben.

Ein Mitglied der Gruppe Super Administrator kann zusätzlich Templates installieren, Systemnachrichten empfangen, Systeminfos abfragen und globalen Checkin durchführen (alle noch nicht veröffentlichten Artikel freigeben).

Für alle diese Nutzergruppen ist eine Registrierung am Content Management System notwendig. Die Nutzer welche nur Besucher auf der Website sind und sich nicht über ein Login personalisieren lassen, werden automatisch als unregistered Users ohne Gruppenzugehörigkeit behandelt.

5.5 Integritätssicherung

Ziel der Integritätssicherung ist die Gewährleistung von Korrektheit und Vollständigkeit der gespeicherten Daten. Dabei wird generell unterschieden in semantische, operationelle und physische Integrität.

5.5.1 Semantische Integrität

Die semantische Integrität gewährleistet die Richtigkeit und Korrektheit der Daten bei jeglicher Nutzereingabe. Dafür können schon direkt beim Datenbankentwurf spezielle Integritätsbedingungen erarbeitet werden. Diese enthalten unter anderem Angaben zu Attributwerten, Spalten, Tupeln oder Relationen auf welche sich die Integritäts-bedingung bezieht, sowie Angaben wann die Einhaltung der Bedingung überprüft werden soll und was bei Verletzung der Bedingung passieren soll.

Einfache Möglichkeiten der Sicherung sind z.B.

- Formatkontrolle von der Webapplikation bei der Eingabe im Webformular wo direkt auf Datentypen und Länge des eingegebenen Datums geprüft wird
- Verhinderung und Zurückweisung von Duplikaten durch Primärschlüssel und UNIQUE-Klauseln beim Datenbankentwurf
- Angabe von NOT NULL bei Primärschlüsseln um undefinierte Werte zu verhindern
- Aufbau von Referentieller Integrität durch Beziehungen zwischen Spalten in verschiedenen Tabellen zur Sicherung der logischen Widerspruchsfreiheit
- Einführung von Daten- und ereignisorientierten Prüfbedingungen durch Trigger und Stored Procedures
- Setzen von Default-Werten bei bestimmten Spalten um keine unbestimmten Werte zuzulassen, z.B.

```
CREATE TABLE element (
    symbol character varying(10) NOT NULL,
    name character varying(50) NOT NULL,
    ...
    stoichiometriccoefficient smallint DEFAULT 0 NOT NULL,
    ...
    dbdatetime timestamp DEFAULT CURRENT_TIMESTAMP,
    atomicnumber numeric NOT NULL
);
```

Referentielle Integrität

Eine Möglichkeit um Inkonsistenzen in einer Datenbank zu vermeiden, ist die Definition von Beziehungen mittels referentieller Integrität. Das bedeutet, dass jedem Fremdschlüssel in einer Tabelle ein entsprechender Primärschlüssel in einer anderen Tabelle zugeordnet ist oder der Wert des Fremdschlüssels NULL ist. Es wird so garantiert, dass der Primärschlüssel in der referenzierten Tabelle existiert und es nicht

zu Inkonsistenzen kommt. Dazu können auch verschiedene Regeln definiert werden, was mit abhängigen Daten beim Ändern oder Löschen passieren soll:

- NO ACTION (default): Die Operation wird durchgeführt und es muss am Ende über Trigger dafür gesorgt werden, dass die Fremdschlüsselbeziehungen nicht verletzt werden.
- RESTRICT: Die Operation wird zurückgewiesen falls ein Primärschlüssel gelöscht wird der noch Abhängigkeiten zu anderen Objekten hat.
- CASCADE: Die Änderungen werden an die abhängigen Objekte weitergegeben. Wenn also ein Primärschlüssel gelöscht wird, so werden bei ON DELETE CASCADE auch alle abhängigen Objekte in der anderen Tabelle gelöscht.
- SET NULL: Falls ein Primärschlüssel gelöscht wird werden die Fremdschlüssel der abhängigen Objekte auf NULL gesetzt
- SET DEFAULT: Hier werden die Werte der abhängigen Fremdschlüssel automatisch auf den Default-Wert der Spalte gesetzt.

Diese Regeln werden hier im Beispiel separat nach der Tabellendefinition festgelegt:

```
CREATE TABLE editor (
    name character varying(50) NOT NULL,
    affiliation character varying(50) NOT NULL,
    email character varying(50)
);
ALTER TABLE ONLY editor
    ADD CONSTRAINT editor_pkey PRIMARY KEY (name);

CREATE TABLE element (
    symbol character varying(10) NOT NULL,
    name character varying(50) NOT NULL,
    molarmass numeric NOT NULL,
    ...
    ...
    editor character varying(50),
    dbdatetime timestamp without time zone DEFAULT now(),
    atomicnumber numeric NOT NULL
);

ALTER TABLE ONLY element
    ADD CONSTRAINT element_pkey PRIMARY KEY (symbol);

ALTER TABLE ONLY element
    ADD CONSTRAINT element_editor_fkey FOREIGN KEY (editor) REFERENCES editor(name)
    ON UPDATE CASCADE ON DELETE RESTRICT;
    ...
    ...
```

Trigger und Stored Procedures

Trigger und Stored Procedures helfen zur Bewahrung der semantischen Integrität. Es sind ereignisgesteuerte Funktionen, die beim Eintreten bestimmter Ereignisse ausgeführt werden. Der Trigger selbst bestimmt nur, bei welchem Ereignis eine Funktion ausgeführt werden soll. Im folgenden Beispiel wird ein Trigger erstellt, der den Namen **tu_pcon** hat und nach einem **UPDATE** in der Tabelle **pcon** für jede Zeile die Funktion **fn_tu_pcon** aufruft. In der Stored Procedure (in PL/pgSQL geschrieben) wird dann beispielsweise geprüft ob der neue geänderte Datensatz in der Spalte **editor** auch wirklich einen Wert aus der Spalte **name** der Tabelle **editor** hat. Ist das nicht der Fall, wird eine Exception mit einer Fehlermeldung hervorgerufen und der Vorgang abgebrochen. Zuletzt wird noch der Wert der Spalte **dbdatetime** auf den aktuellen Zeitstempel (**current_timestamp**) gesetzt.

```
/* Trigger tu_pcon */
CREATE TRIGGER tu_pcon
      BEFORE UPDATE ON pcon
      FOR EACH ROW
      EXECUTE PROCEDURE fn_tu_pcon()

/* Stored Procedure fn_tu_pcon() */
declare
      nRows integer;
      maxCard integer;

begin
 /* Check parent table "Editor", when child table "PCon" changes. */
 if new.Editor != old.Editor then
      select count(*) into nRows
      from Editor
      where new.Editor = Editor.Name;
      if (nRows = 0) then
            raise exception 'Parent does not exist in table "Editor".
Cannot update child table "PCon".';
      end if;
 end if;
 ...
 ...
 /* Update timestamp */
 new.dbdatetime := current_timestamp;
 return new;
end;
```

Trigger und Stored Procedures können aber auch für Logging-Zwecke verwendet werden. Angenommen es sollen alle Veränderungen an Datensätzen einer bestimmten Tabelle in einer weiteren Tabelle geloggt werden, dann wäre ein Trigger die ideale Lösung dafür. Der Trigger wird bei jedem UPDATE, INSERT oder DELETE aufgerufen und die aufgerufene Funktion loggt dann alle Änderungen am Datensatz oder speichert einfach beim UPDATE den vorherigen und den neuen Datensatz bzw. beim INSERT und DELETE den neu eingefügten bzw. alten gelöschten Datensatz in einer neuen Tabelle. Mit diesem Konzept lassen sich auch Zustände von Tabellen zu früheren Zeitpunkten wiederherstellen oder einfach nur alle Änderungen an Datensätzen im Nachhinein überschauen. Das ist im THEREDA-Vorhaben sehr sinnvoll, da viel Wert auf die Zurückverfolgbarkeit von Daten gelegt wird und man auf diese Art einfach prüfen könnte wann welche Änderung an einem bestimmten Datum vorgenommen wurde.

5.5.2 Operationelle Integrität

Unter operationeller Integrität versteht man den Verlust von Daten beim gleichzeitigen Schreiben mehrerer Nutzer zu verhindern bzw. deren Transaktionen möglichst zu Isolieren. Um zu verhindern, dass mehrere Benutzer in einer Transaktion gleichzeitig einen Wert verändern und dann beim COMMIT eine der Schreiboperationen verloren geht, gibt es die Möglichkeit, Sperren zu verwenden. Wird z.B. in zwei Transaktionen derselbe Datensatz geändert, ist es sinnvoll die Zeile in der Tabelle vor der Änderung zu sperren. Die zweite Transaktion wartet dann, bis die Sperre wieder aufgehoben ist und führt erst dann die Änderung durch. Dabei wird unterschieden in Table-Level Locks, welche ganze Tabellen sperren und in Row-Level Locks, wo sich die Sperren nur auf einzelne Zeilen der Tabelle beziehen. In beiden Varianten gibt es unterschiedliche Arten von Sperren, welche zum Teil parallel noch andere Aktionen (z.B. Lesen von Datensätzen) erlauben.

In PostgreSQL werden diese Sperren bei den meisten Datenbankoperationen bereits automatisch gesetzt. Beispielsweise wird bei einem ALTER TABLE Befehl automatisch ein Exclusive Lock auf die gesamte Tabelle gesetzt, das heißt, dass die Tabelle während dessen für alle anderen Operationen gesperrt ist.

Durch solche Sperren kann es allerdings auch passieren, dass zwei Transaktionen sich gegenseitig Sperren und damit vergeblich warten bis die Sperren aufgehoben werden. So eine Situation wird Deadlock genannt. In PostgreSQL werden Deadlocks automatisch erkannt und es wird dann eine der beiden Transaktionen abgebrochen [I_POSTG1].

5.5.3 Physische Integrität

Die physische Integritätssicherung beinhaltet die Wiederherstellung eines integeren Zustandes nach physischen Fehlern wie z.B. beim Hardware-Defekt einer Festplatte oder einem Brand. Am besten geschieht dies durch automatisierte Backups der gesamten Datenbank auf unterschiedlichen physischen und räumlich getrennten Servern. Ein Konzept über verteilte Backups ist bereits im Dokument THEREDA Backup Strategien [S_LESKE1] näher erläutert.

Weiterhin gibt es auch noch die automatische Recovery-Funktion des DBMS selbst. Dabei wird beim Erkennen eines fehlerhaften Zustandes die Kopie eines korrekten Zustandes der Datenbank eingespielt und mit Hilfe der protokollierten Transaktionen in den Log-Dateien ein aktueller korrekter Zustand wiederhergestellt.

Zum Recovery gehört ebenfalls das Rollback bei einer fehlgeschlagenen Transaktion, um danach den ursprünglichen Zustand der Datenbank wieder herzustellen. Dabei werden einfach die Informationen des Logfiles ausgewertet und alle Änderungen bis zum Beginn der Transaktion rückgängig gemacht.

5.5.4 Signieren von Datenbasen für externe Nutzer

Zusätzlich zur Integritätssicherung in der Datenbank selbst soll auch die Integrität der downloadbaren Datenbasen gesichert werden. Um vermeintliche Veränderungen oder Manipulationen zu identifizieren, soll zu jeder Datei eine Prüfsumme separat zum Download angeboten werden. Damit kann im Nachhinein zweifelsfrei bestätigt oder widerlegt werden, ob ein Benutzer mit den originalen, unveränderten Daten von THEREDA gearbeitet hat.

Es gibt viele verschiedene Prüfsummenverfahren, aber die Einfacheren laufen generell nach dem gleichen Schema ab: Grundlegende Komponenten der Daten wie Bits / Bytes werden mit einem bestimmten Faktor multipliziert und dann der Reihe nach aufsummiert. Das daraus resultierende Ergebnis wird Prüfsumme genannt. Der Empfänger kann dann diese Summe aus seinen eigenen Daten erneut berechnen und durch einen Vergleich verifizieren ob seine Daten authentisch sind.

Das einfachste Beispiel dafür ist die Quersumme, wo aber z.B. Vertauschungen von Zahlen nicht erkannt werden können.

Da die einfachen Prüfsummen nur vor unbeabsichtigten Veränderungen schützen können, jedoch bei absichtlicher Manipulation leicht umgangen werden können, ist es notwendig, noch stärkere kryptografische Algorithmen zu verwenden. Dazu zählen unter anderem die Einweg-Hash-Funktionen, welche auch große Sicherheit vor Manipulation bieten. Die am häufigsten verwendete kryptographische Hashfunktion für diesen Zweck ist die Message-Digest Algorithm 5 (MD5) - Funktion.

Bei der heutigen Technik ist es aber schon möglich mit überschaubarem Aufwand auch unterschiedliche Nachrichten zu erzeugen welche die gleiche Prüfsumme aufweisen. Zur einfachen Verifizierung von heruntergeladenen Dateien ist es aber dennoch ausreichend. Im speziellen Fall von THEREDA ist zu berücksichtigen, dass Textänderungen zur Erzeugung der gleichen Prüfsumme nicht beliebig möglich sind, da der Text und die Zahlen selbst im sinnvollen Kontext zueinander stehen müssen.

Falls doch hundertprozentige Sicherheit gegen Manipulation gewährleistet werden soll, ist laut aktuellem Stand der Technik eine SHA-256-Prüfsumme erforderlich. Diese wird momentan als "sicher" angesehen.

5.5.5 Verifikation von vorhandenen Datenbasen

Zur späteren Verifikation von vorhandenen Datenbasen auf Benutzerseite genügt es, mit Hilfe von Programmen wie QuickSFV [I_QUICK1] oder FastSum [I_FASTS1] die MD5-Summe aus der vorhandenen Datei erneut zu berechnen und einfach mit der auf der THEREDA-Website zur Verfügung gestellten originalen Prüfsumme zu vergleichen. Sollten dabei Abweichungen auftreten, so sind Veränderungen an den Daten vorgenommen worden und die Echtheit der Daten kann nicht garantiert werden.

6. Prototypische Implementierung einer Joomla! - Komponente

6.1 Joomla!-Komponenten und das MVC-Konzept

Joomla! hat im Sinne der Ordnung in Version 1.5 das Model-View-Controller (MVC) Konzept eingeführt. Dieses Konzept ist in der Softwareentwicklung weit verbreitet um komplexe Anwendungen übersichtlich und gut gegliedert zu halten und Veränderungen bzw. Erweiterungen für andere Entwickler möglichst einfach zu gestalten.

Es wird bei den Aufgaben einer Software fast immer von drei Bereichen gesprochen:

- Ein Datenmodell (Model)
- Eine Präsentation (View)
- Eine Programmsteuerung (Controller)

Das Model enthält die darzustellenden Daten wobei die Herkunft dieser nicht relevant ist. Es ist auch dort keine Information über die Ausgabe der Daten bekannt. Die Präsentation stellt die Daten des Modells dar, wobei aber eine Verbindung zum Modell vorhanden sein muss, damit überhaupt etwas dargestellt werden kann. Und der Controller steuert die Modelle und Präsentationen und reagiert auf Benutzereingaben und andere Ereignisse und gibt Informationen an die Präsentation weiter.

Das MVC ist ein Konzept um Ordnung in einem ständig wachsenden System zu halten. In Joomla! 1.0 existierte dieses Konzept noch nicht und jeder konnte nach freier Verfügung programmieren. Durch das MVC-Konzept ist es in Version 1.5 einfacher Komponenten und Funktionalitäten den eigenen Wünschen anzupassen oder zu erweitern.

Im Sequenzdiagramm in **Abbildung 9** wird der schematische Ablauf einer HTTP-Anfrage dargestellt. Dabei wird zuerst der Controller aufgerufen, welcher entscheidet, welches Model und welche View verwendet werden. Zur Datenbeschaffung hat der Controller auch Zugriff auf das Model. Dann wird vom Controller je nach Aktion die entsprechende View aufgerufen. In dieser werden die anzuzeigenden Daten vom Model abgefragt und mit Hilfe des gewählten Templates an den Browser gesendet und angezeigt.

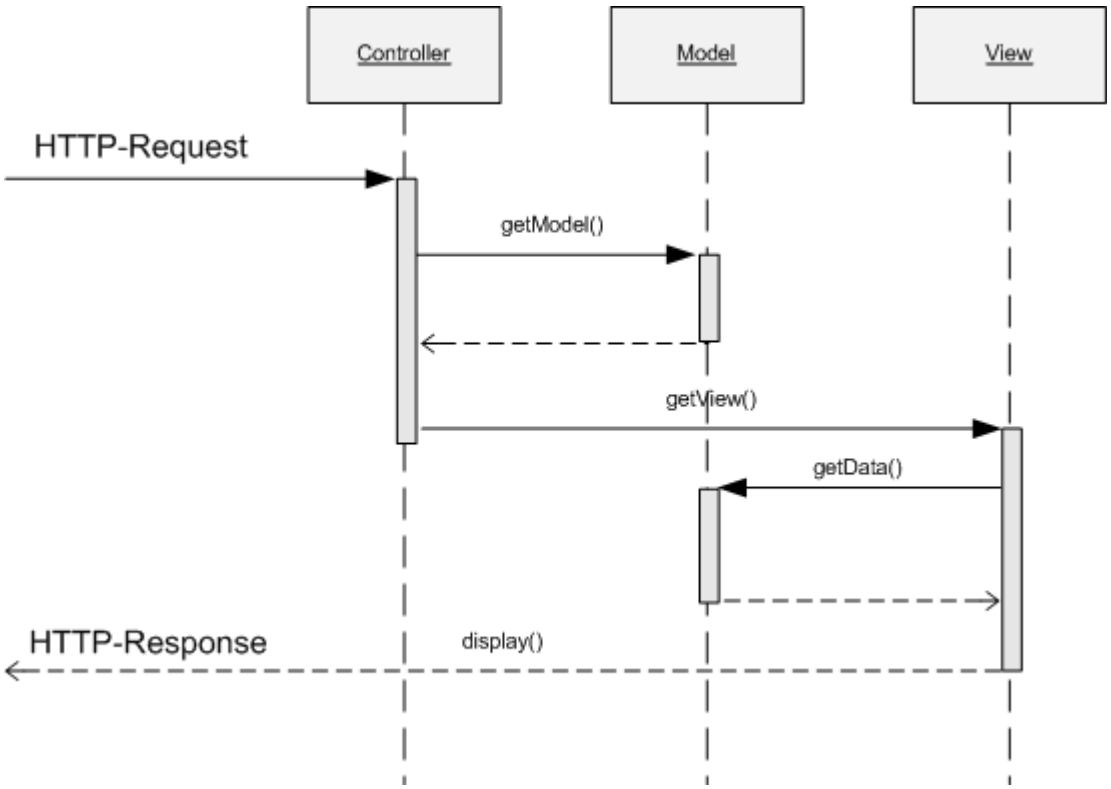


Abbildung 9 - Sequenzdiagramm: HTTP-Anfrage an eine Website mit MVC-Konzept

6.2 Dateien/Struktur der Komponente

6.2.1 Frontend

Da Joomla! ein mächtiges Framework bereitstellt, ergeben sich damit auch vorgegebene Strukturen für Komponenten bzw. folgende Dateistruktur für das Frontend der THEREDA-Komponente:

Sämtliche Code-Dateien zur THEREDA-Komponente befinden sich vom Joomla!-Root aus in **`/components/com_thereda`**. Die Übersetzungen für die Textbausteine hingegen sind separat für alle Komponenten in **`/languages`** abgelegt. Dort gibt es je ein Verzeichnis für jede Sprache in der dann in .ini-Dateien die Textbausteine übersetzt gespeichert sind. Hier sind also nur die Dateien

`/languages/de-DE/de-DE.com_thereda.ini` und

`/languages/en-GB/en-GB.com_thereda.ini`

von Bedeutung. Die Auswahl und das Laden der entsprechenden Dateien übernimmt das Joomla!-Framework.

Die eigentlichen Code-Dateien der Komponente sind nochmals in mehrere Bereiche und Verzeichnisse gegliedert. Wie im Abschnitt 6.1 erwähnt, wird in Model (Datenhaltung), View (Anzeige) und Controller (Logik) unterschieden. Die Models werden im Komponentenverzeichnis unter **`models`** abgelegt und nach den Klassennamen der dazugehörigen View benannt. Der Hauptcontroller liegt direkt im Hauptverzeichnis (**`controller.php`**) und falls benötigt können weitere Controller im Unterordner **`controllers`** abgelegt werden.

Für jede View gibt es ein extra Verzeichnis mit einer php-Datei für die Steuerung und das Laden/Verarbeiten der Daten (meist **`view.html.php`**) sowie einem Ordner **`tmpl`**, wo das Template dazu abgelegt ist (meist **`default.php`**).

Desweiteren gibt es noch den Ordner **`tables`**, wo Klassen für den Tabellenzugriff auf Datenbanken abgelegt werden und einen Ordner **`helpers`**, wo Hilfs-Klassen abgelegt werden, die in mehreren Views Verwendung finden.

In **Abbildung 10** wird ein Teil der Datei- und Verzeichnisstruktur des Frontends als Baum dargestellt.

```
|   controller.php              - der Hauptcontroller
|   thereda.php                 - für Aufrufen von Controller
|
|---helpers                     - (helpers-Verzeichnis)
|   |---single.php              - Helper-Klasse für Einzeldatenabfrage
|   |
|---tables                      - (helpers-Verzeichnis)
|   |---singlesession.php       - Table-Klasse für Session (Einzeld.)
|   |
|---models                      - (models-Verzeichnis)
|   |---single.php              - Model für 1. View Einzeldatenabfrage
|   |---single2.php             - Model für 2. View Einzeldatenabfrage
|   |---single3.php             - Model für 3. View Einzeldatenabfrage
|   |
|---images                          - Bilder für Einzeldatenabfrage
|   |---temp[0-14].png              - Bilder für Temperaturanzeige
|   |
|---controllers                     - (controller-Verzeichnis)
|   |
|---views                           - (views-Verzeichnis)
|   |   |---single
|   |   |   |---view.html.php       - (view single)
|   |   |   |---tmpl/default.php    - (template single)
|   |   |---single2
|   |   |   |---view.html.php       - (view single2)
|   |   |   |---tmpl/default.php    - (template single2)
|   |   |---single3
|   |   |   |---view.html.php       - (view single3)
|   |   |   |---view.raw.php        - (view single3-export ohne Layout)
|   |   |   |---tmpl/default.php    - (template single 3 default)
|   |   |   |---tmpl/exportcsv.php  - (template single 3 export csv)
|   |   |   |---tmpl/exportxls.php  - (template single 3 export xls)
|   |   |---...
|   |   |   |---...
|   |   |   |---...
```

Abbildung 10 - Datei- und Verzeichnisstruktur der THEREDA-Komponente (Frontend)

6.2.2 Backend

Das Backend ist analog zum Frontend genauso strukturiert, da es auch nach dem MVC-Konzept aufgebaut ist. Der einzige Unterschied ist, dass die Dateien der Komponente im Backend in dem Verzeichnis **`/administrator/components/com_thereda`** liegen.

Weiterhin sind hier auch noch Dateien abgelegt die für die Komponente sowohl im Backend als auch im Frontend benötigt werden, wie zum Beispiel die Datenbank-Klasse für PostgreSQL oder die Singleton-Klasse zur Vorhaltung der Konfigurationsparameter, welche unter **`/administrator/components/com_thereda/classes`** abgelegt sind.

In **Abbildung 11** die Datei- und Verzeichnisstruktur des Backends als Baum dargestellt.

```
|   controller.php              - der Hauptcontroller
|   admin.thereda.php           - für Aufrufen vom Controller
|   install.sql                 - SQL-Befehle für Tabellenerzeugung bei
|                                 Installation
|   uninstall.sql               - SQL-Befehle für Deinstallation
|   thereda.xml                 - für Installationsparameter,
|                                 Komponenteninformationen, Version und
|                                 zugehörige Dateien
|---tables                      - (helpers-Verzeichnis)
|   |---config.php              - Table-Klasse für Config-Parameter
|   |
|---models                      - (models-Verzeichnis)
|   |---config.php              - Model für Anzeige der Konf.-Parameter
|   |---config_edit.php         - Model für Bearbeitung der Konf.-Param.
|   |
|---controllers                     - (controller-Verzeichnis)
|   |
|---views                           - (views-Verzeichnis)
|   |   |---config
|   |   |   |---view.html.php       - (view config)
|   |   |   |---tmpl/default.php    - (template config)
|   |   |---config_edit
|   |   |   |---view.html.php       - (view config_edit)
|   |   |   |---tmpl/default.php    - (template config_edit)
```

Abbildung 11 - Datei- und Verzeichnisstruktur der THEREDA-Komponente (Backend)

6.2.3 Ablage der THEREDA-Dateien (Datenbasen)

Die vordefinierten Datenbasen werden direkt in einem Unterverzeichnis des Webservers in einer flachen Struktur abgelegt. Für die Benennung der Dateien wurde folgendes Schema festgelegt (mit einer konstanten Gesamtlänge von 10+1+3+1+4+1+7 = 27 Zeichen):

`Datum_Wechselwirkungsmodell_Datenformat.thereda`

Das Datum wird im Format nach ISO 8601/EN 28601 angegeben, also in der Form 2010-01-18.

Als Wechselwirkungsmodelle stehen das Extended Debye-Hückel-Modell (EDH), das Pitzer-Modell (PIT) und das Specific Ion Interaction - Modell (SIT) zur Verfügung, also jeweils ein Kürzel aus drei Buchstaben. Für die Varianten des Datenformates sind JSON, EQ36, PHRQ, GCWB und CAPP geplant, hier also jeweils mit vier Buchstaben abgekürzt. Die Dateiendung wird mit .thereda festgelegt.

Daraus ergeben sich für eine Datenbasis vom 1. April 2010 zum Beispiel folgende Dateinamen (wobei bisher nicht alle Varianten sinnvoll sind):

```
2010-04-01_EDH_JSON.thereda
2010-04-01_EDH_EQ36.thereda
2010-04-01_EDH_PHRQ.thereda
2010-04-01_EDH_GCWB.thereda
2010-04-01_EDH_CAPP.thereda
2010-04-01_SIT_JSON.thereda
2010-04-01_SIT_EQ36.thereda
2010-04-01_SIT_PHRQ.thereda
2010-04-01_SIT_GCWB.thereda
2010-04-01_SIT_CAPP.thereda
2010-04-01_PIT_JSON.thereda
2010-04-01_PIT_EQ36.thereda
2010-04-01_PIT_PHRQ.thereda
2010-04-01_PIT_GCWB.thereda
2010-04-01_PIT_CAPP.thereda
```

Diese Dateien sollen von außen nicht direkt zugreifbar sein, sondern nur für registrierte Benutzer über den Menüpunkt THEREDA Datenabfrage -> Vordefinierte DB heruntergeladen werden können.

6.3 Datenbankstruktur der Komponente

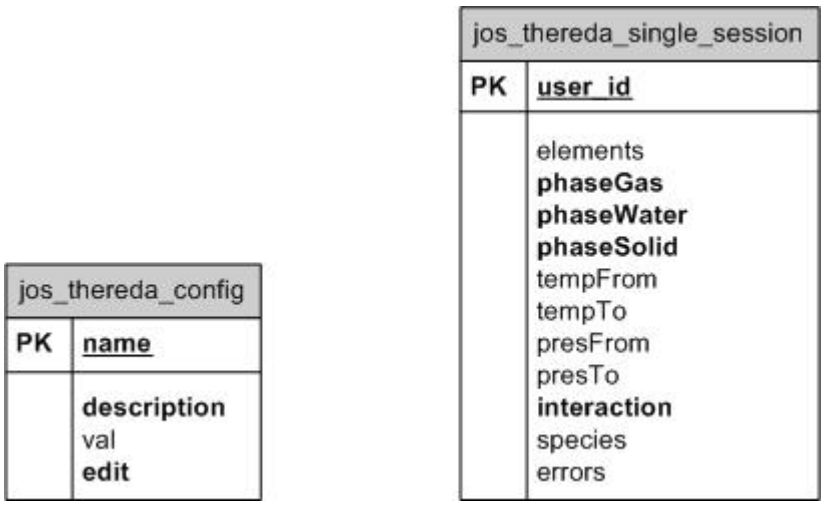


Abbildung 12 - Datenbankstruktur der Joomla!-Komponente (MySQL)

Tabelle jos_thereda_config

In dieser Tabelle werden alle Konfigurationsparameter gespeichert, welche dann auch innerhalb der Komponente über eine Singelton-Klasse abrufbar sind. D.h. es existiert nur genau eine Instanz dieser Klasse in der Anwendung, welche außerdem global verfügbar ist. Da die Parameter nicht nach Spalten definiert sind, ist die Konfiguration jederzeit problemlos erweiterbar (siehe **Abb. 13**).

name	description	val	edit
pg_host	PostgreSQL-Server Hostname	localhost	1
pg_user	PostgreSQL-Server Username		1
pg_pass	PostgreSQL-Server Password		1
pg_db	PostgreSQL-Server Database	thereda	1
admin_email	Email for Error-Reporting	access@thereda.de	1
pear	path to pear classes	/usr/share/php	1

Abbildung 13 - Joomla!-Tabelle (MySQL) für Konfigurationsparameter

Tabelle jos_thereda_single_session

In dieser Tabelle wird für jeden Benutzer eine Session der Einzeldatenabfrage vorgehalten. Das beinhaltet alle gewählten Einstellungen, Filter und Eingaben für Formulare wie die Auswahl des Elements, Phasentyp, Temperaturbereich, Wechselwirkungsmodell und gewählte Spezies.

6.4 Implementierung einzelner Anforderungen

6.4.1 Verknüpfung von Joomla! mit PostgreSQL

Bevor im Einzelnen auf die Anforderungen eingegangen wird, sind zuerst ein paar Grundlagen zu schaffen um zum Beispiel überhaupt mit Hilfe des Joomla!-Frameworks auf die PostgreSQL-Datenbank zugreifen zu können. Joomla! unterstützt nativ bisher nur MySQL als Datenbanksystem was die Nutzung der vom Framework bereitgestellten Funktionalität etwas einschränkt.

Über die Klasse JFactory wird in Joomla! eine Reihe von wichtigen Basisklassen initialisiert. Die Instanzen dieser sind damit alle über das Singleton-Entwurfsmuster abgesichert. Dasselbe gilt auch für die Datenbankklasse JDatabase [B_KEMPK1]. Die Instanz dieser Klasse ist in der Anwendung global verfügbar und kann wie im folgenden Beispiel einfach über die get-Methode **`JFactory::getDBO()`** geholt werden:

```
<?
	$db =& JFactory::getDBO();
	$query = 'SELECT * FROM #__thereda_config';
	$db->setQuery( $query );
	$result = $db->loadresult();
?>
```

Diese Instanz die man bei dem obigen Befehl bekommt, ist aber eine Instanz der Klasse JDatabaseMysql, welche von JDatabase erbt und entsprechende SQL-Befehle

überschreibt. Die Funktionalität ist daher nur für eine MySQL-Datenbank verwendbar. Die eigentliche THEREDA-Datenbank ist jedoch in einem PostgreSQL-DBS untergebracht. Dafür musste wie in Abschnitt 5.2.2 beschrieben eine neue Klasse JDatabasePgSQL implementiert werden, welche auch alle Parameter und Methoden von JDatabase erbt die entsprechenden Funktionen zur Kommunikation mit der Datenbank für PostgreSQL überschreibt. Diese Klasse ist für die Joomla!-Komponente in

`/administrator/components/com_thereda/classes/pgsql.class.php`

abgelegt.

Eine Instanz ist über eine selbst implementierte Singleton-Klasse mit Namen Thereda abrufbar, welche außerdem über die Methode **`getConfig()`** die Konfigurationsparameter aus der MySQL-Datenbank bereit stellen kann:

```
<?
	$db = Thereda::getDBO();
	$query = 'SELECT pcon, element
		  FROM pconcomposition
		  ORDER BY pcon, element';
	$db->setQuery( $query );
	$result = $db->loadObjectList();
	$pearPath = Thereda::getConfig('pear');
?>
```

In Joomla! gibt es weiter die Möglichkeit sogenannte Helper-Klassen zu implementieren, die dann in allen Views und Modellen einer Komponente verwendet werden können. In der THEREDA-Komponente gibt es beispielsweise die Klasse TheredaHelper, welche für die Einzeldatenabfrage in allen Modellen die vorgehaltene Session mit allen Eingabeparametern bereit stellt.

6.4.2 Einzeldatenabfrage

Nachdem der Nutzer sich im Joomla!-Frontend eingeloggt hat und sich im Menü der Einzeldatenabfrage befindet, erscheint je nach Nutzergruppe ein Formular wo ein Element per Radiobutton ausgewählt und noch weitere Spezifika zur Filterung eingegeben werden können (**Abb. 14**). Unter anderem kann noch der Phasentyp (Gas, gelöste Spezies und feste Phase), ein Temperaturbereich, für den die gefundenen Daten gültig sein sollen, und ein Wechselwirkungsmodell gewählt werden. Die Auswahl eines Druckbereiches ist bisher nur für die Zukunft vorgesehen.

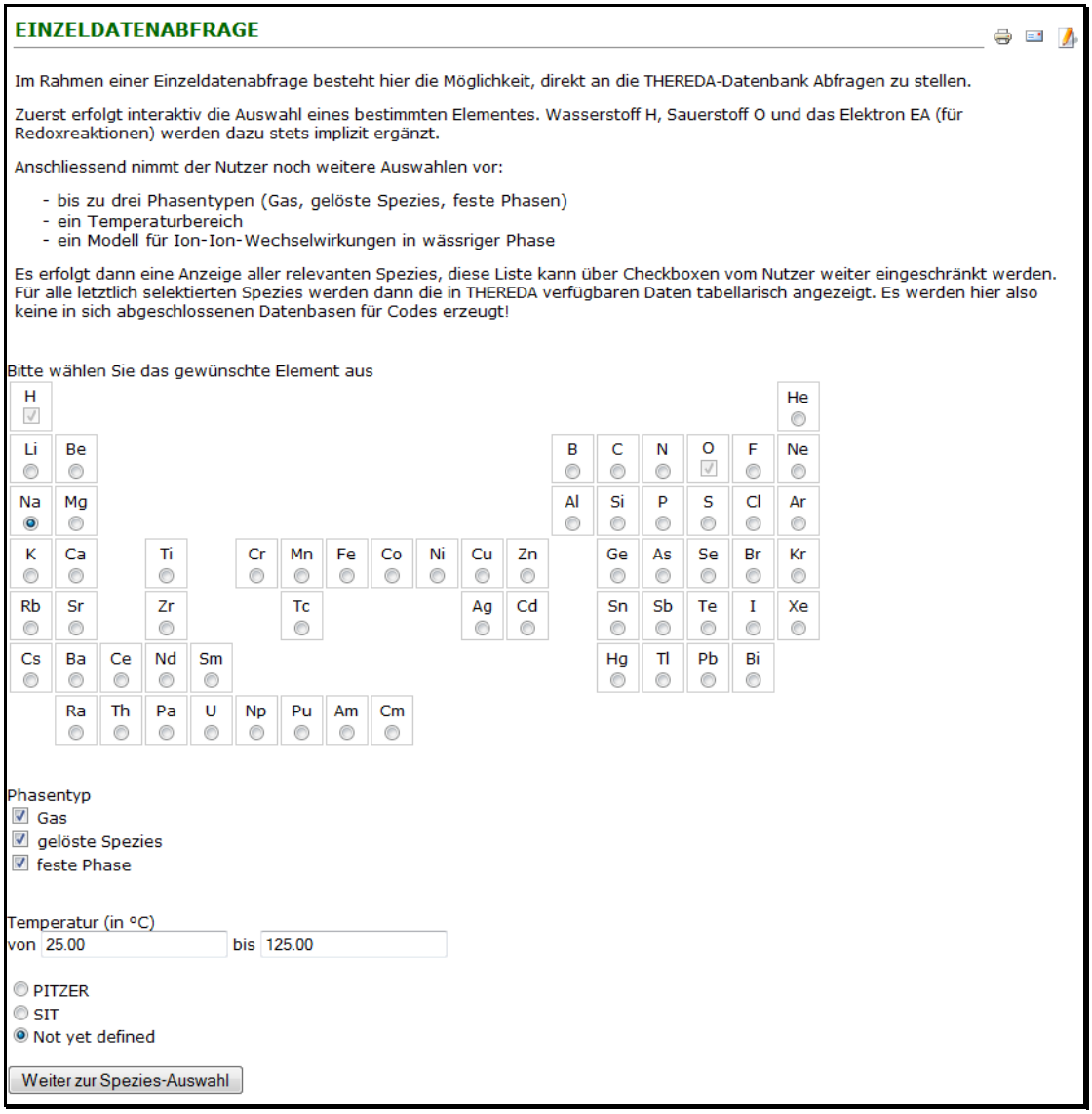
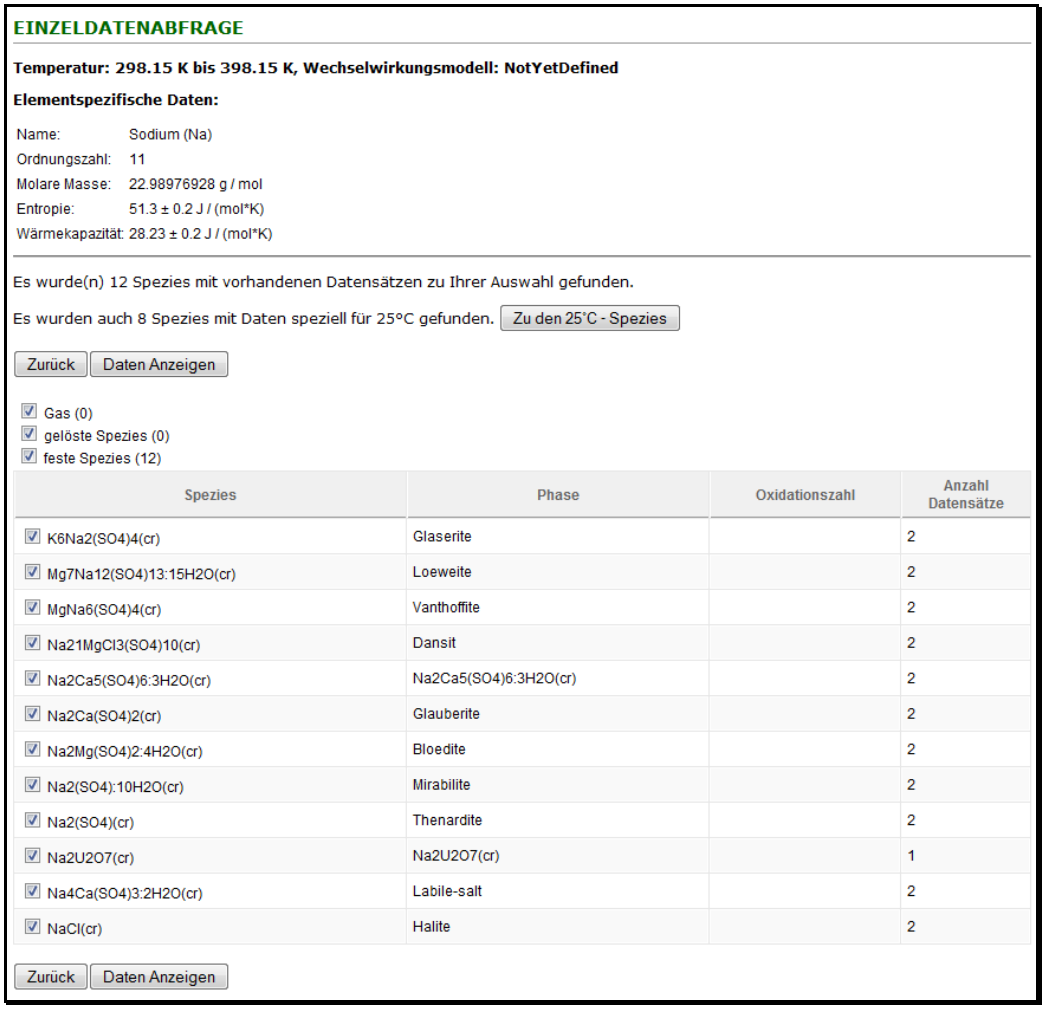


Abbildung 14 - Screenshot: Einzeldatenabfrage - Auswahlkriterien

Nach dem Absenden des Formulars werden über den Controller der Komponente diese Daten in einer Session gespeichert und der Nutzer wird bei gültiger Eingabe zur Auswahl der Spezies weitergeleitet (**Abb. 15**). Es werden alle Spezies angezeigt, die Datensätze mit der gewünschten Eingrenzung in Temperatur, Phasentyp und Wechselwirkungsmodell beinhalten. Der Nutzer hat hier noch die Möglichkeit direkt mit einem Klick alle Spezies eines Phasentyps, einer Oxidationszahl oder auch einzelne Spezies aus- oder abzuwählen. Außerdem wird die Anzahl der vorhandenen Datensätze mit angezeigt. Falls ein Temperaturbereich gewählt wurde, welcher 25°C einschließt, wird zusätzlich ein Verweis auf eventuell weitere Datensätze genau für 25°C ausgegeben. Die Auswahlkriterien und elementspezifischen Daten werden immer oben auf der Seite angezeigt, unabhängig davon, ob zur gewünschten Auswahl Datensätze vorhanden sind.



EINZELDATENABFRAGE

Temperatur: 298.15 K bis 398.15 K, Wechselwirkungsmodell: NotYetDefined

Elementspezifische Daten:

Name:	Sodium (Na)
Ordnungszahl:	11
Molare Masse:	22.98976928 g / mol
Entropie:	51.3 ± 0.2 J / (mol*K)
Wärmekapazität:	28.23 ± 0.2 J / (mol*K)

Es wurde(n) 12 Spezies mit vorhandenen Datensätzen zu Ihrer Auswahl gefunden.

Es wurden auch 8 Spezies mit Daten speziell für 25°C gefunden. Zu den 25°C - Spezies

Zurück Daten Anzeigen

☑ Gas (0)
☑ gelöste Spezies (0)
☑ feste Spezies (12)

Spezies	Phase	Oxidationszahl	Anzahl Datensätze
☑ K6Na2(SO4)4(cr)	Glaserite		2
☑ Mg7Na12(SO4)13:15H2O(cr)	Loeweite		2
☑ MgNa6(SO4)4(cr)	Vanthoffite		2
☑ Na21MgCl3(SO4)10(cr)	Dansit		2
☑ Na2Ca5(SO4)6:3H2O(cr)	Na2Ca5(SO4)6:3H2O(cr)		2
☑ Na2Ca(SO4)2(cr)	Glauberite		2
☑ Na2Mg(SO4)2:4H2O(cr)	Bloedite		2
☑ Na2(SO4):10H2O(cr)	Mirabilite		2
☑ Na2(SO4)(cr)	Thenardite		2
☑ Na2U2O7(cr)	Na2U2O7(cr)		1
☑ Na4Ca(SO4)3:2H2O(cr)	Labile-salt		2
☑ NaCl(cr)	Halite		2

Zurück Daten Anzeigen

Abbildung 15 - Screenshot: Einzeldatenabfrage - Speziesauswahl

Nach der Spezies-Auswahl werden die gefundenen Daten in tabellarischer Übersicht aufgelistet und die Möglichkeit zum Export als CSV oder MS Excel-Datei angeboten. Die Unterscheidung wurde für notwendig erachtet, da MS Excel sich bei CSV-Dateien nicht an den allgemeinen Dateiaufbau laut RFC 4180 hält und diese somit fehlinterpretiert.

Je nachdem, ob ein Temperaturbereich oder nur Daten für 25°C gewählt wurden, stehen in der Ausgabetabelle Temperaturfunktionen und zugehörige Koeffizienten oder nur feste Werte und deren Fehler. Bei den Daten für einen Temperaturbereich gibt es festgelegte Limits in denen die Temperaturfunktion gültig ist. Um diese anschaulich mit dem nutzerspezifizierten Bereich (farblich) darzustellen, wurde eine Programmlogik implementiert, die alle auftretenden Fälle von Überschneidungen und Übereinstimmungen abdeckt (siehe **Tab. 2**).

Tabelle 2 - Programmlogik für die Darstellung der Beziehung zwischen Temperaturlimits in THEREDA und dem nutzerspezifizierten Temperaturbereich

tf<mintk	tf=mintk	tf>mintk	tf<maxtk	tf=maxtk	tf>maxtk	tt<mintk	tt=mintk	tt>mintk	tt<maxtk	tt=maxtk	tt>maxtk	tf=tt	Bild
		x	x					x	x				1
	x		x					x		x			7
x			x					x			x		2
x			x				x		x				5
x			x					x	x				3
	x		x					x	x				8
		x	x					x		x			9
		x	x					x			x		4
		x		x				x			x		6
x			x					x		x			14
	x		x					x			x		13
	x		x				x		x			x	10
		x	x					x	x			x	11
		x		x				x		x		x	12

Legende:

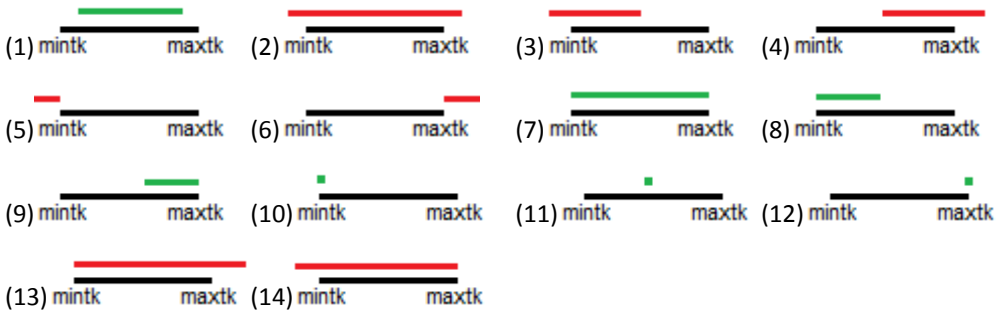
tf: Anfang des nutzerspezifizierten Temperaturbereiches

tt: Ende des nutzerspezifizierten Temperaturbereiches

mintk: Anfang des Gültigkeitsbereiches der Temperaturfunktion

maxtk: Ende des Gültigkeitsbereiches der Temperaturfunktion

Bilder:



6.4.3 Komplexe Systeme

Der Bereich der komplexen Systeme ist als Webapplikation von der Firma Lineas bereits implementiert worden. Die Authentifizierung über Joomla! wurde nachträglich noch hinzugefügt (siehe Abschnitt 5.2.1). Jetzt gilt es nur noch die Einbettung in die THEREDA-Website vorzunehmen.

Man könnte die Webapplikation direkt über den Wrapper von Joomla! einbinden. Das hätte aber den Nachteil, dass nicht ohne Weiteres für jede Benutzergruppe der Zugriff separat freigegeben werden kann und es wäre auch nicht möglich einen einfach zu verwaltenden Einleitungstext vor dem Auswahlformular auszugeben.

Aus diesen Gründen wurde für die Java Webapplikation innerhalb der THEREDA-Komponente noch eine zusätzliche View angelegt, die im Menüpunkt "Komplexe Systeme" verlinkt ist. In dieser View wird ein einfacher Joomla! - Artikel als Einleitungstext vorgeschaltet und die Anwendung darunter über ein IFrame eingebettet. Da die Webapplikation derzeit nur als Testversion für die Joomla!-Gruppe Autoren verfügbar sein sollte, wurde innerhalb der View eine entsprechende Unterscheidung getroffen, dass für die einfachen registrierten Nutzer nur ein Hinweistext erscheint und für die Autoren und höheren Gruppen die Applikation angezeigt wird. Der Einleitungstext ist, da es ein einfacher Joomla! - Artikel ist, an das Layout angepasst und ohne Probleme mit den von Joomla! angebotenen Mitteln zu editieren oder in die englische Sprache zu übersetzen.

6.4.4 Vordefinierte Datenbasen

Die vordefinierten Datenbasen sollen ohne großen Aufwand eingepflegt werden können. Sie werden in einem separaten Verzeichnis auf dem Webserver abgelegt, auf welches der direkte Zugriff von außerhalb mittels einer .htaccess - Datei geschützt ist. Die Datenbasen werden von einem Projektmitarbeiter erzeugt und manuell auf den

Server in das Verzeichnis hochgeladen. Die Komponente prüft dann beim Aufrufen der Seite ob bereits Dateien mit Dateinamen im vorgegebenen Format (siehe 6.2.3) vorhanden sind. Falls nicht bereits erstellt wird dann automatisch eine Datei mit der MD5-Prüfsumme für jede der Datenbasen erzeugt. Auf der Übersichtsseite werden die vorhandenen Datenbasen zum Download angeboten. Die zugehörigen MD5-Summen werden neben dem Dateinamen angezeigt oder als separate Datei zum Download angeboten.

Da die Datenbasen nicht über einen normalen Downloadlink direkt erreichbar sind (weil das Verzeichnis für den externen Zugriff gesperrt ist) müssen sie über ein PHP-Skript durchgereicht werden. Das hat den Vorteil, dass vor jedem Download nochmals die Authentizität des Nutzers überprüft und bei Bedarf protokolliert werden kann.

So müssen zum Bereitstellen neuer Datenbasen lediglich die Dateien auf den THEREDA-Server hochgeladen und die Konventionen für die Dateinamen eingehalten werden.

6.5 Installation / Deinstallation der Komponente

Jede Joomla!-Komponente ist zum Installieren normalerweise in ein Zip-File gepackt und beinhaltet ein XML-File, welches alle Daten zur Installation und Deinstallation beinhaltet.

In dieser Datei stehen allgemeine Komponenteninformationen wie Autor, Name, Copyright und eine Liste aller unterstützten Sprachen und deren zugehörige Sprachdateien. Weiterhin sind auch alle anderen Dateien, die für den Front- und Backend-Bereich vorhanden sind, mit aufgeführt.

Für sonstige Initialisierungen kann auch ein PHP-Skript zum installieren und deinstallieren angegeben werden, wo evtl. noch bestimmte Routinen abgearbeitet werden können.

Zur Initialisierung/Löschung der Datenbank kann zusätzlich noch ein SQL-Skript angegeben werden um z.B. alle benötigten Tabellen zu erstellen und diese mit vordefinierten Standardwerten zu initialisierten.

Beim Deinstallations-Skript muss genau überlegt werden, ob wirklich alle vorhandenen Daten und Tabellen gelöscht werden sollen. Bei einem Update der Komponente kann es zum Beispiel notwendig sein, die alte Komponente erst zu entfernen, die Daten jedoch beizubehalten und im Installationsskript der neuen Komponente nur die Datenstruktur anzupassen.

Das Selbe gilt natürlich auch für die angelegten Datenbasen und alle anderen Daten, die in Form von Dateien auf dem Webserver von der Komponente abgelegt wurden.

Im Thereda-Vorhaben hat sich das Team darauf geeinigt bei der Deinstallation der Komponente vorerst alle vorhandenen Daten im System beizubehalten.

7. Fazit / Ausblick

Mit dieser Diplomarbeit wurde ein Konzept zur Definition heterogener Schnittstellen zur THEREDA-Datenbank mit Berücksichtigung verschiedener funktionaler und qualitativer Anforderungen erstellt. Dieses Konzept ist zum Teil bereits prototypisch realisiert.

Die Anforderungen in Bezug auf die Einzeldatenabfrage wurden für die thermodynamischen Daten (mit konstant 25°C und mit Temperaturfunktion) vollständig realisiert. Die Abfrage für Wechselwirkungskoeffizienten und der Download von vordefinierten Datenbasen ist teilweise implementiert und wird voraussichtlich in den kommenden Wochen fertiggestellt. Im April 2010 sollen bereits die ersten vorgefertigten Datenbasen aus THEREDA veröffentlich werden.

Die Idee der Umsetzung als Joomla!-Komponente hat sich für die Integration in das bisherige System als sehr vorteilhaft herausgestellt. Die bereits entwickelte Java-Applikation für den Export der JSON-Dateien war hingegen schwieriger mit der bisherigen Internetpräsenz zu vereinen, da eine unabhängig laufende Anwendung einer anderen Programmiersprache nur schwer in ein anderes System integriert werden kann. Es ist daher allgemein nicht sinnvoll, bei der Entwicklung von Webapplikationen zweigleisig mit unterschiedlichen Umgebungen (Java und PHP) zu arbeiten. Sinnvoller in Bezug auf Integration, Programmieraufwand und Sicherheit wäre es, alle Anwendungen in einer Programmiersprache bzw. einem System zu entwickeln. Da die bisherige Internetpräsenz mit Joomla! komplett in PHP geschrieben ist und so auch im Laufe der letzten Jahre gewachsen ist, wäre es sinnvoll dies beizubehalten.

Erweiterungen dieser Arbeit wären beispielsweise die Umstellung auf Joomla! 1.6, wenn dies offiziell erschienen ist, da dort eine erweiterte Benutzerverwaltung mit eigenen Gruppen- und Rechtevergaben möglich ist. Dieses Feature würde sehr für eine

Umstellung sprechen, da man in der aktuellen Joomla! - Version mit den sechs vordefinierten Benutzergruppen bereits an die Grenzen stößt.

Ein anderer Punkt für zukünftige Entwicklungen wäre noch die Einarbeitung des am Ende von Abschnitt 5.5.1 angesprochenen Logging von Veränderungen an den Datentabellen in einer separaten Tabelle. Das würde einen weiteren großen Fortschritt zur Datensicherheit und Rückverfolgbarkeit mit sich bringen.

Es wäre auch sinnvoll, die Dateneingabe und Datenpflege als Joomla!-Komponente zu implementieren, da diese in Zukunft nicht mehr über die Synchronisation mit einem MS Excel Dokument, sondern auch über die Weboberfläche abgewickelt werden soll. Dies bringt außerdem im Vergleich zur aktuellen Variante weitaus mehr Funktionalität und vor allem Qualitätssicherung der Daten mit sich.

Zum Abschluss kann man sagen, dass das hier entwickelte Konzept für Schnittstellen für heterogene Zugangssysteme einen entscheidenden Baustein und eine gute Vorlage für die Weiterentwicklung von THEREDA darstellt. Diese Diplomarbeit mit der Realisierung der Schnittstellen hat in der bisherigen Projektlaufbahn eine erste Möglichkeit geschaffen, die Daten der THEREDA-Datenbank anderen externen Nutzern auf einer einfachen, sicheren und übersichtlichen Art und Weise zur Verfügung zu stellen. Der bisherige Prototyp ist schon als Testversion im Einsatz und hat bisher positives Feedback von anderen Projekt-Mitarbeitern bekommen. Er wird in der nächsten Projektphase noch vervollständigt und erweitert werden, um alle gewünschten Anforderungen zu realisieren.

Literaturverzeichnis

Anmerkung: Das Literaturverzeichnis ist alphabetisch nach Bereichen geordnet. Alle Internetseiten wurden am 23.02.2010 vor Drucklegung der Diplomarbeit auf Gültigkeit und Erreichbarkeit geprüft.

Bücher [B_xxxxx9]

[B_BETHK1] Bethke, C. M., Geochemical Reaction Modeling, Oxford University Press, 1996

[B_EBERS1] Ebersbach, A. , Glaser, M. , Kubani, R.: Joomla! 1.5: Das umfassende Handbuch, Galileo Press, 2008, 2. Auflage

[B_KEMPK1] Kempkens, A.: Das Joomla!-Entwicklerhandbuch, Addison-Wesley, 2009, 1. Auflage

[B_MATTH1] Matthew, N., Stones R.: Beginning Databases with PostgreSQL - From Novice to Prefessional, Apress, 2005

[B_MOEHR1] Möhrke, C.: Besser PHP programmieren - Handbuch professioneller PHP-Techniken, Galileo Computing, 2009, 3. Auflage

[B_SCHER1] Scherbaum A.: PostgreSQL. Datenbankpraxis für Anwender, Administratoren und Entwickler, Open Source Press, 2009, 1. Auflage

[B_VOSSE1] Vossen, G.: Datenmodelle, Datenbanksprachen und Datanbankmanagementsysteme, Oldenbourg Verlag, 2008, 5. Auflage

Internet [I_xxxxx9]

[I_FASTS1] FastSum - MD5 checksum software for Windows, Kirill Zinov
Link: http://www.fastsum.com
Zuletzt geprüft: 23.02.2010

[I_JSON1] Einführung in JSON, (unbekannt)
Link: http://json.org/json-de.html
Zuletzt geprüft: 23.02.2010

[I_POSTG1] PostgreSQL 8.3.9 Documentation, PostgreSQL Global Development Group
Link: http://www.postgresql.org/docs/8.3/interactive/explicit-locking.html
Zuletzt geprüft: 23.02.2010

[I_QUICK1] Official QuickSFV Homepage, (unbekannt)
Link: http://www.quicksfv.org
Zuletzt geprüft: 23.02.2010

[I_SUNJD1] Sun JDBC Drivers, Sun Developer Network
Link: http://www.fastsum.com
Zuletzt geprüft: 23.02.2010

[I_THERE1] Internetauftritt THEREDA Database Project
Link: http://www.thereda.de
Zuletzt geprüft: 23.02.2010

Normen und Standards

[ISO 8601]/ [EN 28601] ISO 8601 / EN 28601: Standard / Europäische Norm zur Beschreibung numerischer Datumsformate und Zeitangaben

[ANSI SQL1] ANSI SQL1 1986: SQL-Spezifikation von ANSI

[ISO SQL1] ISO SQL1 1987: SQL-Spezifikation von ISO

[ISO SQL2/ SQL-92] ISO SQL2/SQL-92 1992: SQL-Spezifikation von ISO

[ISO SQL3/ SQL:1999] ISO SQL3/SQL:1999 1999: SQL-Spezifikation von ISO

[ISO SQL:2003] ISO SQL:2003 2003: SQL-Spezifikation von ISO

[RFC 4180] RFC-Spezifikation des Dateiformates CSV

Sonstige [S_xxxxx9]

[S_GRAEF1] Graefe, G., Toll, A.: Datenbanksysteme: Skript zur Lehrveranstaltung, HTW Dresden, SS 2009

[S_HERZO1] Herzog, M.: Entwurf und prototypische Realisierung von Maßnahmen eines Autorisierungs- und Datensicherheitskonzeptes in einer SQL-basierten chemischen Stoffdatenbank, HTW Dresden, WS 2009

[S_LESKE1] Leske, S.: THEREDA Backupstrategien, Forschungszentrum Dresden - Rossendorf, 2009

Zeitschriften und Reports [Z_xxxxx9]

[Z_ALTMA1] Altmaier, M., Brendler, V., Hagemann, S., Herbert, H.-J., Kienzler, B., Marquardt, C.M., Moog, H.C., Neck, V., Richter, A., Voigt, W., Wilhelm, S.: "THEREDA – Ein Beitrag zur Langzeitsicherheit von Endlagern nuklearer und nichtnuklearer Abfälle", atw - Internationale Zeitschrift für Kernenergie, 53, 2008

[Z_ERIKS1] Eriksson, G., Spencer, P. J., Sippola, H., A General Thermodynamic Software Interface, Proceedings 2nd Colloquium on Process Simulation, Helsinki University of Technology, 1995, 113.

[Z_PARKH1] Parkhurst, D.L., 1995, User's guide to PHREEQC--A computer program for speciation, reaction-path, advective-transport, and inverse geochemical calculations: U.S. Geological Survey Water-Resources Investigations Report 95-4227, 143 p.

[Z_THERE1] THEREDA-Team, THEREDA Abschlussbericht Rev. 1.0, 12/2009

[Z_WOLER1] Wolery, T. J. UCRL-MA-110662 Part I. Lawrence Livermore National Laboratory, California, USA, 1992

Anhang

Anlage 1 - JSON - Dateistrukturdefinition

```
#############################################################################
# JSON Template für THEREDA
#
# Version vom 28-August-2009
#
# Generelles
#
#  Identische Benennung der Felder in JSON und PostgreSQL erleichtert Abfragen.
#  Remarks werden nie exportiert.
#  Editoren werden nie exportiert.
#  Alternates werden vorerst noch nicht exportiert.
#
# Der Anwender kann auswählen zwischen zwei Wechselwirkungsmodellen:
# "Pitzer" und "SIT". Entsprechend müssen die Daten ausgelesen werden
# aus data_standard_pitzer, data_variable_pitzer und data_standard_sit
# (Eine Tabelle data_variable_sit gibt es noch nicht).
#
# Teilweise sind nachfolgend fiktive Daten eingetragen, da noch keine
# entsprechenden Werte in THEREDA vorhanden sind. Manche Blöcke sind auch
# leer, da deren Struktur bereits zuvor ausführlich dokumentiert wurde.

{ ## Hier beginnt das Objekt der kompletten THEREDA-DB im JSON-Format

#############################################################################
# Es folgen alle chemischen Elemente, zu denen THEREDA Daten kennt.
# (inklusive dem Elektron EA)

        "Elements": ## Erstes Objekt: Chemische Elemente
        [ ## Hier beginnt die Liste aller Elemente
                { ## Hier beginnt das Objekt für das erste Element
                        "symbol": "H",
                        "name": "Hydrogen",
                        "molarmass": 1.00794,
                        "molarmass_referenceid": "WIE2006",
                        "s298": 130.68,
                        "s298_unctype": "Gauss2s",
                        "s298_negativeunc": 0.003,
                        "s298_positiveunc": 0.003,
                        "s298_referenceid": "COX/WAG1989",
                        "cp298": 28.836,
                        "cp298_unctype": "Gauss2s",
                        "cp298_negativeunc": 0.002,
                        "cp298_positiveunc": 0.002,
                        "cp298_referenceid": "COX/WAG1989",
                        "referencestate": "gaseous",
                        "stoichiometriccoefficient": 2,
                        "description": "1st example for element"
                }, ## Ende des Objektes für das erste Element
                { ## Hier beginnt das Objekt für das nächste Element
                ## ...
                } ## Ende der Daten für das zweite Element, weitere würden folgen
        ], ## Ende der Liste aller Elemente

#############################################################################
# Es folgen alle Phasen.
# Jede Phase ist ein Objekt und besteht aus 1 ... k Konstituenten,
# welche genau einem aus vier verschiedenen Typen zugeordnet sind:
# - PrimaryMaster
# - Secondarymaster
# - MineralSolid
# - Product
#
```

```
# Es ist THEREDA-Konvention, dass alle Phasen in folgender Reihenfolge
# abgearbeitet werden:
# GAS
# - PrimaryMaster: enthält vorerst nur H2(g)
# - SecondaryMaster: enthält vorerst nur O2(g)
# - MineralSolid (leer, da in der Gasphase nicht definiert)
# - Product
# AQUEOUS
# - PrimaryMaster
# - SecondaryMaster
# - MineralSolid (leer, da in der wässrigen Phase nicht definiert)
# - Product
# MIXED-PHASE 1 (feste Mischphase)
# - PrimaryMaster (leer, da in Festphasen nicht definiert)
# - SecondaryMaster (leer, da in Festphasen nicht definiert)
# - MineralSolid
# - Product (leer, da in Festphasen nicht definiert)
# ...
# MIXED-PHASE n (feste Mischphase)
# - PrimaryMaster (leer, da in Festphasen nicht definiert)
# - SecondaryMaster (leer, da in Festphasen nicht definiert)
# - MineralSolid
# - Product (leer, da in Festphasen nicht definiert)
# PHASE-(2+n+1) (Reiner Feststoff)
# - PrimaryMaster (leer, da in Festphasen nicht definiert)
# - SecondaryMaster (leer, da in Festphasen nicht definiert)
# - MineralSolid
# - Product (leer, da in Festphasen nicht definiert)
# ...
# PHASE-(2+n+m) (Reiner Feststoff)
# - PrimaryMaster (leer, da in Festphasen nicht definiert)
# - SecondaryMaster (leer, da in Festphasen nicht definiert)
# - MineralSolid
# - Product (leer, da in Festphasen nicht definiert)
#
# Innerhalb jedes Phasenkonstituententyps gibt es mehrere Phasenkonstituenten.
# Für jeden werden verschiedene Datenblöcke abgelegt,
# welche jeweils wiederum Objekte sind:
# - Deklaration
# - Zusammensetzung
# - (Bildungs-)Reaktion
# - Standarddaten
# - p,T-Funktionen

        "Phases": ## Zweites Objekt: Phasen
        [ ## Hier beginnt die Liste aller (2+n+m) Phasen
                { ## Hier beginnt das Objekt für die Gasphase
                        "symbol": "g",
                        "modification": "NA",
                        "mixedphase": true,
                        "description": "",
                        "PrimaryMaster":
                        [ ## Hier beginnt die Liste der PrimaryMaster
                                { ## Hier beginnt der bisher einzige Datensatz für H2(g)
                                }
                        ],
                        "SecondaryMaster":
                        [ ## Hier beginnt die Liste der SecondaryMaster
                                { ## Hier beginnt der bisher einzige Datensatz für O2(g)
                                }
                        ],
                        ## In der Gasphase gibt es keine MineralSolids
                        "MineralSolids": [ ],
                        "Product":
                        [ ## Hier beginnt die Liste aller Produktspezies
                                { ## Hier beginnt der Datensatz für CO2(g)
                                        "symbol": "CO2(g)",
                                        "Declaration":
                                        {
                                                "equilibrium_constraint": "Complete
equilibrium",
```

```
                "charge": 0,
                "molarmass": 44.0095,
                "centralelement": "",
                "oxidationnumber": "",
                "redox": false,
                "description": ""
        },
        "Composition":
        [
                {
                        "element": "C",
                        "numberofelement": 1
                },
                {
                        "element": "O",
                        "numberofelement": 2
                }
        ],
        "FormingReaction":
        [
                {
                        "pcon_reactant": "CO2(g)",
                        "coefficient": 1
                },
                {
                        "pcon_reactant": "CO3<2->",
                        "coefficient": -1
                },
                {
                        "pcon_reactant": "H<+>",
                        "coefficient": -2
                },
                {
                        "pcon_reactant": "H2O(l)",
                        "coefficient": 1
                }
        ],
        "DataStandard":
        ## Die Reihenfolge der Datentypen sollte stets
gleich sein:
        ## DFG, DFH, S, CP, V, DRG, DRH, DRS, DRCP, LOGK
        ## In der Gasphase gibt es kein V298.
        ## Nicht für alle Kombinationen aus PCon und
Datentyp
        ## existieren Einträge in der Datenbank!
        [
                {
                        "datatype": "DFG298",
                        "value": -394372.54795,
                        "calcmode": "CGHF",
                        "unctype": "Gauss2s",
                        "negativeunc": 133,
                        "positiveunc": 133,
                        "dataclass": -1,
                        "category": "F",
                        "dataquality": -1,
                        "datasource": -1,
                        "value_notallowed": "",
                        "unctype_notallowed": "",
                        "negativeunc_notallowed": "",
                        "positiveunc_notallowed": "",
                        "reference": "InternallyCalculated",
                        ## Hier und bei allen weiteren
Literaturreferenzen
                        ## ist der Eintrag dem Feld
"referenceid_1" zu entnehmen
                        "description": "",
                        "controllingresult": "Not yet
controlled"
                },
                {
```

```
                "datatype": "DFH298",
                "value": -393510,
                "calcmode": "Entered",
                "unctype": "Gauss2s",
                "negativeunc": 130,
                "positiveunc": 130,
                "dataclass": 1,
                "category": "F",
                "dataquality": 1,
                "datasource": 1,
                "value_notallowed": "",
                "unctype_notallowed": "",
                "negativeunc_notallowed": "",
                "positiveunc_notallowed": "",
                "reference": "GUI/FAN2003",
                "description": "",
                "controllingresult": "Not yet
controlled"
            },
            {
                "datatype": "S298",
                "value": 213.785,
                "calcmode": "Entered",
                "unctype": "Gauss2s",
                "negativeunc": 0.01,
                "positiveunc": 0.01,
                "dataclass": 1,
                "category": "F",
                "dataquality": 1,
                "datasource": 1,
                "value_notallowed": "",
                "unctype_notallowed": "",
                "negativeunc_notallowed": "",
                "positiveunc_notallowed": "",
                "reference": "GUI/FAN2003",
                "description": "",
                "controllingresult": "Not yet
controlled"
            },
            {
                "datatype": "CP298",
                "value": 37.135,
                "calcmode": "Entered",
                "unctype": "Gauss2s",
                "negativeunc": 0.002,
                "positiveunc": 0.002,
                "dataclass": 1,
                "category": "F",
                "dataquality": 1,
                "datasource": 1,
                "value_notallowed": "",
                "unctype_notallowed": "",
                "negativeunc_notallowed": "",
                "positiveunc_notallowed": "",
                "reference": "GUI/FAN2003",
                "description": "",
                "controllingresult": "Not yet
controlled"
            },
            {
                "datatype": "DRG298",
                "value": -103629.503071,
                "calcmode": "CRLOGK",
                "unctype": "Gauss2s",
                "negativeunc": 199,
                "positiveunc": 199,
                "dataclass": -1,
                "category": "R",
                "dataquality": -1,
                "datasource": -1,
                "value_notallowed": "",
```

```
                                        "unctype_notallowed": "",
                                        "negativeunc_notallowed": "",
                                        "positiveunc_notallowed": "",
                                        "reference": "InternallyCalculated",
                                        "description": "",
                                        "controllingresult": "Not yet
controlled"
                                },
                                {
                                        "datatype": "LOGK298",
                                        "value": 18.155,
                                        "calcmode": "Entered",
                                        "unctype": "Gauss2s",
                                        "negativeunc": 0.035,
                                        "positiveunc": 0.035,
                                        "dataclass": 1,
                                        "category": "R",
                                        "dataquality": 1,
                                        "datasource": 1,
                                        "value_notallowed": "",
                                        "unctype_notallowed": "",
                                        "negativeunc_notallowed": "",
                                        "positiveunc_notallowed": "",
                                        "reference": "GUI/FAN2003",
                                        "description": "",
                                        "controllingresult": "Not yet
controlled"
                                }
                        ], ## Ende der Standard-Daten bei 298.15 K
                        "DataVariable": ## fiktive Daten zur
Temperaturabhängigkeit
                        ## Die Reihenfolge der Datentypen sollte stets
gleich sein:
                        ## DFGT, DFHT, CPT, DRGT, DRHT, DRST, DRCPT, LOGKT
                        ## Nicht für alle Kombinationen aus PCon und
Datentyp
                        ## existieren Einträge in der Datenbank!
                        [
                                {
                                        "datatype": "DRGT",
                                        "calcmode": "Entered",
                                        "tpfunc": "NEA-extended",
                                        "a": -1.00239384,
                                        "b": 0,
                                        "c": 0,
                                        "d": 0,
                                        "e": 0.001,
                                        "f": 0,
                                        "mintk": 298.15,
                                        "maxtk": 393.15,
                                        "minpbar": 1.01325,
                                        "maxpbar": 1.01325,
                                        "dataclass": 1,
                                        "category": "R",
                                        "dataquality": 1,
                                        "datasource": 6,
                                        "reference": "irgendeine Referenz",
                                        "description": "",
                                        "controllingresult": "Not yet
controlled"
                                },
                                {
                                        "datatype": "LOGKT",
                                        "calcmode": "CTPFUNC",
                                        "tpfunc": "NEA-extended",
                                        "a": -15.669997,
                                        "b": 0,
                                        "c": 5429.392703,
                                        "d": 0,
                                        "e": 0,
                                        "f": 0,
```

```
                                                "mintk": 298.15,
                                                "maxtk": 393.15,
                                                "minpbar": 1.01325,
                                                "maxpbar": 1.01325,
                                                "dataclass": 1,
                                                "category": "R",
                                                "dataquality": 1,
                                                "datasource": 6,
                                                "reference": "InternallyCalculated",
                                                "description": "",
                                                "controllingresult": "Not yet
controlled"
                                        }
                                ] ## Ende der variablen Daten
                        } ## Ende des Datensatzes für CO2(g), weitere Gase würden
folgen.
                ] ## Ende der Liste der Produktspezies der Gasphase
        }, ## Ende des Objektes für die Gasphase

        { ## Hier beginnt das Objekt für die wässrige Phase
                "symbol": "aq",
                "modification": "NA",
                "mixedphase": true,
                "description": "",
                "PrimaryMaster":
                [ ## Hier beginnt die Liste aller PrimaryMaster von "aq"
                        { ## Hier beginnnt der Datensatz für Na<+>
                                "symbol": "Na<+>",
                                "Declaration":
                                {
                                        "equilibrium_constraint": "Complete
equilibrium",
                                        ## Für PrimaryMaster ist "Complete
equilibrium" vorgegeben
                                        "charge": 1,
                                        "molarmass": 22.98922,
                                        "centralelement": "Na",
                                        "oxidationnumber": 1,
                                        "redox": false,
                                        "description": ""
                                },
                                "Composition":
                                [
                                        {
                                                "element": "Na",
                                                "numberofelement": 1
                                        },
                                        {
                                                "element": "EA",
                                                "numberofelement": -1
                                        }
                                ],
                                "FormingReaction": [ ],
                                ## Entfällt für PrimaryMaster per definitionem,
daher leere Menge
                                ## Für PrimaryMaster entfallen auch alle
reaktionsbezogenen Datentypen
                                "DataStandard":
                                ## Im Gegensatz zur Gasphase gibt es aber hier
V298!
                                [
                                        {
                                                "datatype": "DFG298",
                                                "value": -261952.8935,
                                                "calcmode": "CGHF",
                                                "unctype": "",
                                                "negativeunc": 0,
                                                "positiveunc": 0,
                                                "dataclass": -1,
                                                "category": "F",
                                                "dataquality": -1,
```

```
                                        "datasource": -1,
                                        "value_notallowed": "",
                                        "unctype_notallowed": "",
                                        "negativeunc_notallowed": "",
                                        "positiveunc_notallowed": "",
                                        "reference": "InternallyCalculated",
                                        "description": "",
                                        "controllingresult": "Not yet
controlled"
                                },
                                {
                                        "datatype": "DFH298",
                                        "value": -240340,
                                        "calcmode": "Entered",
                                        "unctype": "Gauss2s",
                                        "negativeunc": 60,
                                        "positiveunc": 60,
                                        "dataclass": 1,
                                        "category": "F",
                                        "dataquality": 1,
                                        "datasource": 1,
                                        "value_notallowed": "",
                                        "unctype_notallowed": "",
                                        "negativeunc_notallowed": "",
                                        "positiveunc_notallowed": "",
                                        "reference": "GUI/FAN2003",
                                        "description": "",
                                        "controllingresult": "Not yet
controlled"
                                },
                                {
                                        "datatype": "S298",
                                        "value": 58.45,
                                        "calcmode": "Entered",
                                        "unctype": "Gauss2s",
                                        "negativeunc": 0.15,
                                        "positiveunc": 0.15,
                                        "dataclass": 1,
                                        "category": "F",
                                        "dataquality": 1,
                                        "datasource": 1,
                                        "value_notallowed": "",
                                        "unctype_notallowed": "",
                                        "negativeunc_notallowed": "",
                                        "positiveunc_notallowed": "",
                                        "reference": "GUI/FAN2003",
                                        "description": "",
                                        "controllingresult": "Not yet
controlled"
                                }
                        ],
                        "DataVariable": ## fiktive Daten
                        [
                                {
                                        "datatype": "LOGKT",
                                        "calcmode": "CTPFUNC",
                                        "tpfunc": "NEA-transformed",
                                        "a": 292.79995,
                                        "b": 0.008153,
                                        "c": 0.000028,
                                        "d": -17704.717064,
                                        "e": 1520656.906219,
                                        "f": -42.380672,
                                        "mintk": 273.15,
                                        "maxtk": 523.15,
                                        "minpbar": 1.01325,
                                        "maxpbar": 1.01325,
                                        "dataclass": 1,
                                        "category": "R",
                                        "dataquality": 1,
                                        "datasource": 3,
```

```
                                                "reference": "InternallyCalculated",
                                                "description": "",
                                                "controllingresult": "Not yet
controlled"
                                        },
                                        {
                                        ## ...
                                        }
                                ]
                        }       ## Ende des Datensatzes für Na<+>, weitere
PrimaryMaster folgen.
                ], ## Ende der Liste der Primary Masters in "aq"
                "SecondaryMaster":
                [ ## Hier beginnt die Liste der Secondary Masters in "aq"
                        { ## Hier beginnt der Datensatz für ClO3<->
                                "symbol": "ClO3<->",
                                "Declaration":
                                {
                                        "equilibrium_constraint": "Complete
equilibrium",
                                        "charge": -1,
                                        "molarmass": 83.451749,
                                        "centralelement": "Cl",
                                        "oxidationnumber": 5,
                                        "redox": true,
                                        "description": ""
                                },
                                "Composition":
                                [
                                        {
                                                "element": "Cl",
                                                "numberofelement": 1
                                        },
                                        {
                                                "element": "O",
                                                "numberofelement": 3
                                        },
                                        {
                                                "element": "EA",
                                                "numberofelement": 1
                                        }
                                ],
                                "FormingReaction":
                                [
                                        {
                                                "pcon_reactant": "ClO3<->",
                                                "coefficient": 1
                                        },
                                        {
                                                "pcon_reactant": "Cl<->",
                                                "coefficient": -1
                                        },
                                        {
                                                "pcon_reactant": "H2(g)",
                                                "coefficient": 3
                                        },
                                        {
                                                "pcon_reactant": "H2O(l)",
                                                "coefficient": -3
                                        }
                                ],
                                "DataStandard":
                                [
                                ## ...
                                ],
                                "DataVariable":
                                [
                                ## ...
                                ]
                        } ## Ende des Datensatzes für ClO3<->, weitere
SecondaryMaster folgen.
```

```
                    ], ## Ende der Liste für Secondary Master in "aq"
                    "MineralSolids": [ ], ## in "aq" nicht definiert
                    "Product":
                    [ ## Hier beginnt die Liste der Product Species in "aq"
                           { ## Hier beginnt der Datensatz für HCO3<->
                                  "symbol": "HCO3<->",
                                  "Declaration":
                                  {
                                         "equilibrium_constraint": "Complete
equilibrium",
                                         "charge": -1,
                                         "molarmass": 61.017389,
                                         "centralelement": "C",
                                         "oxidationnumber": 4,
                                         "redox": false,
                                         "description": ""
                                  },
                                  "Composition":
                                  [
                                         {
                                                "element": "H",
                                                "numberofelement": 1
                                         },
                                         {
                                                "element": "C",
                                                "numberofelement": 1
                                         },
                                         {
                                                "element": "O",
                                                "numberofelement": 3
                                         },
                                         {
                                                "element": "EA",
                                                "numberofelement": 1
                                         }
                                  ],
                                  "FormingReaction":
                                  [
                                         {
                                                "pcon_reactant": "HCO3<->",
                                                "coefficient": 1
                                         },
                                         {
                                                "pcon_reactant": "CO3<2->",
                                                "coefficient": -1
                                         },
                                         {
                                                "pcon_reactant": "H<+>",
                                                "coefficient": -1
                                         }
                                  ],
                                  "DataStandard":
                                  [
                                  ## ...
                                  ],
                                  "DataVariable":
                                  [
                                  ## ...
                                  ]
                           } ## Ende des Datensatzes für HCO3<->, weitere Products
würden folgen
                    ] ## Ende der Liste für Products in "aq"
             }, ## Ende des Objektes der wässrige Phase

             { ## Hier beginnt das Objekt der ersten festen Mischphase: "Olivine"
                    ## Jede feste Mischphase besteht aus mindestens zwei sogenannten
                    ## Endgliedern, welche MineralSolids sind (also reine Feststoffe)
                    ## Achtung: in der Datenbank gibt es derzeit noch keine festen
                    ## Mischphasen! Das folgende dient daher nur als Beispiel:
                    "symbol": "Olivine",
                    "modification": "cr",
```

```
                    "mixedphase": true,
                    "description": "",
                    "PrimaryMaster": [ ],
                    "SecondaryMaster": [ ],
                    "MineralSolids":
                    [
                           { ## Hier beginnt der Datensatz für 1. Endglied FeSiO3(cr)
                                  "Symbol": "FeSiO3",
                                  "Declaration":
                                  {
                                         "equilibrium_constraint": "Complete
equilibrium",
                                         "charge": 0,
                                         "molarmass": 131.9288,
                                         "centralelement": "Fe",
                                         "oxidationnumber": 2,
                                         "redox": false,
                                         "description": "",
                                  },
                                  "Composition":
                                  [
                                         {
                                                "element": "Fe",
                                                "numberofelement": 1
                                         },
                                         {
                                                "element": "Si",
                                                "numberofelement": 1
                                         },
                                         {
                                                "element": "O",
                                                "numberofelement": 3
                                         }
                                  ],
                                  "FormingReaction":
                                  [
                                         {
                                                "pcon_reactant": "",
                                                "coefficient": ""
                                         }
                                  ],
                                  "DataStandard":
                                  [
                                  ## ...
                                  ],
                                  "DataVariable":
                                  [
                                  ## ...
                                  ]
                           }, ## Hier endet der Datensatz für 1. Endglied FeSiO3(cr) =
Fayalit
                           { ## Hier käme der Datensatz für 2. Endglied MgSiO3(cr) =
Forsterit
                           }
                    ],     ## Ende der Liste der MineralSolids von "Olivine"
                    "Products": [ ]
             }, ## Ende des Objektes "Olivine" (die erste feste Lösung)

             ## Es folgt eine beliebige Anzahl weiterer fester Mischphasen, immer
             ## in derselben Struktur wie oben beim Olivin gezeigt,
             ## auch mit mehr Endgliedern.
             { ## Hier beginnt irgendeine weitere Mischphase
                    ## ...
             }, ## Ende der weiteren Mischphase

             ## Nun folgen all jene (stöchiometrisch reinen) Festphasen
             ## welche nur aus genau einem Endglied bestehen.

             { ## Hier beginnt die erste reine Festphase "Halit"
                    "symbol": "Halit",
                    "modification": "cr",
```

```
                    "mixedphase": false,
                    "description": "",
                    "PrimaryMaster": [ ],
                    "SecondaryMaster": [ ],
                    "MineralSolids":
                    [ ## Diese Klammerebene ist nur formal, da Halit ja nur ein PCon
hat
                            { ## Öffnende Klammer für NaCl(cr)
                                    "symbol": "NaCl(cr)",
                                    "Declaration":
                                    {
                                            "equilibrium_constraint": "Complete
equilibrium",
                                            "charge": 0,
                                            "molarmass": 58.442769,
                                            "centralelement": "",
                                            "oxidationnumber": "",
                                            "redox": false,
                                            "description": ""
                                    },
                                    "Composition":
                                    [
                                            {
                                                    "element": "Na",
                                                    "numberofelement": 1
                                            },
                                            {
                                                    "element": "Cl",
                                                    "numberofelement": 1
                                            }
                                    ],
                                    "FormingReaction":
                                    [
                                            {
                                                    "pcon_reactant": "NaCl(cr)",
                                                    "coefficient": 1
                                            },
                                            {
                                                    "pcon_reactant": "Na<+>",
                                                    "coefficient": -1
                                            },
                                            {
                                                    "pcon_reactant": "Cl<->",
                                                    "coefficient": -1
                                            }
                                    ],
                                    "DataStandard":
                                    [
                                    ## ...
                                    ],
                                    "DataVariable":
                                    [
                                            {
                                                    "datatype": "LOGKT",
                                                    "calcmode": "CTPFUNC",
                                                    "tpfunc": "NEA-transformed",
                                                    "a": -10515.677962,
                                                    "b": -3.487404,
                                                    "c": 0.001258,
                                                    "d": 412401.165236,
                                                    "e": -17729301.586319,
                                                    "f": 1801.021055,
                                                    "mintk": 298.15,
                                                    "maxtk": 393.15,
                                                    "minpbar": 1.01325,
                                                    "maxpbar": 1.01325,
                                                    "dataclass": -1,
                                                    "category": "R",
                                                    "dataquality": -1,
                                                    "datasource": -1,
                                                    "reference": "InternallyCalculated",
```

```
                                                "description": "",
                                                "controllingresult": "Not yet
controlled"
                                        },
                                        {
                                                "datatype": "DRGT",
                                                "calcmode": "Entered",
                                                "tpfunc": "NEA-extended",
                                                "a": 7895365.369,
                                                "b": -201321.2537,
                                                "c": 34480.30816,
                                                "d": -66.76588747,
                                                "e": 0.02407524316,
                                                "f": -339425117,
                                                "mintk": 298.15,
                                                "maxtk": 393.15,
                                                "minpbar": 1.01325,
                                                "maxpbar": 1.01325,
                                                "dataclass": 1,
                                                "category": "R",
                                                "dataquality": 1,
                                                "datasource": 6,
                                                "reference": "NotYetDetermined",
                                                "description": "",
                                                "controllingresult": "Not yet
controlled"
                                        }
                                ]
                        } ## Hier endet der Datensatz für NaCl(cr)
                ],      ## Ende der Liste der MineralSoids für Halite
                "Products": [ ]
        } ## Ende des Objektes für Halite, weitere reine Festphasen würden folgen
], ## Ende der Liste aller Phasen überhaupt

########################################################################
# Es folgen alle Wechselwirkungen (WW) mit Definition und Parametern
# Die WW sind je nach Phase (Gas, Wasser, Mischkristall) mit völlig
# unterschiedlichen Modellen beschreibbar, aber die Struktur für WW
# ist so generisch, dass alle in das gleiche Muster fallen.
#
# Für jede Mischphase kann und muss jeweils nur genau ein Modell
# der Wechselwirkungen vorselektiert werden.
# Für reine Phasen gibt es keine WW.

"InteractionPhases": ## Drittes Objekt: Phasen mit WW
[ ## Hier beginnt die Liste aller Mischphasen mit WW
        { ## Hier beginnt das Objekt der WW in der Phase "aq"
                "phase": "aq",
                "interactionmodel": "Pitzer",
                ## Achtung: In einer JSON-Datei kann für die wässrige Phase nur
                ## entweder Pitzer oder SIT auftaucht. Lediglich zu illustrativen
                ## Zwecken snd in diesem Template beide Modelle simultan
vertreten.
                "interaction":
                [ ## Hier beginnt die Liste aller Pitzer-WW (binär und ternär)
                        { ## Hier beginnt der Datensatz der ersten (binären) WW
                                "Declaration":
                                {
                                        "interactiontype": "Pitzer_binary",
                                        "pcon_1": "K<+>",
                                        "pcon_2": "Cl<->",
                                        "pcon_3": "",
                                        "description": ""
                                },
                                "IP298":
                                {
                                        "ip298_1": 0.0480802587884002,
                                        "ip298_1_unctype": "",
                                        "ip298_1_negativeunc": 0,
                                        "ip298_1_positiveunc": 0,
                                        "ip298_2": 0.218076817366779,
```

```
        "ip298_2_unctype": "",
        "ip298_2_negativeunc": 0,
        "ip298_2_positiveunc": 0,
        "ip298_3": 0,
        "ip298_3_unctype": "",
        "ip298_3_negativeunc": 0,
        "ip298_3_positiveunc": 0,
        "ip298_4": -0.000787989096430533,
        "ip298_4_unctype": "",
        "ip298_4_negativeunc": 0,
        "ip298_4_positiveunc": 0,
        "ip298_5": 2,
        "ip298_5_unctype": "",
        "ip298_5_negativeunc": 0,
        "ip298_5_positiveunc": 0,
        "ip298_6": 0,
        "ip298_6_unctype": "",
        "ip298_6_negativeunc": 0,
        "ip298_6_positiveunc": 0,
        "calcmode": "CTPFUNC",
        "dataquality": -1,
        "datasource": -1,
        "reference": "InternallyCalculated",
        "description": ""
},
"IPT":
{
        "tpfunc": "Pitzer-function",
        "ip_1_a": -758.476330506946,
        "ip_1_b": 26.7372347224728,
        "ip_1_c": -4.70618514763365,
        "ip_1_d": 0.0100719838607252,
        "ip_1_e": -0.00000375989815382765,
        "ip_1_f": 0,
        "ip_2_a": 112193.168416622,
        "ip_2_b": -2804.03483552829,
        "ip_2_c": 478.32163208852,
        "ip_2_d": -0.907183089061278,
        "ip_2_e": 0.000323929103974983,
        "ip_2_f": -4946661.79806362,
        "ip_3_a": 0,
        "ip_3_b": 0,
        "ip_3_c": 0,
        "ip_3_d": 0,
        "ip_3_e": 0,
        "ip_3_f": 0,
        "ip_4_a": 91.2701122617115,
        "ip_4_b": -3.30527358590414,
        "ip_4_c": 0.586443127668531,
        "ip_4_d": -0.00129806286728005,
        "ip_4_e": 0.000000495707610920681,
        "ip_4_f": 0,
        "ip_5_a": 0,
        "ip_5_b": 2,
        "ip_5_c": 0,
        "ip_5_d": 0,
        "ip_5_e": 0,
        "ip_5_f": 0,
        "ip_6_a": 0,
        "ip_6_b": 0,
        "ip_6_c": 0,
        "ip_6_d": 0,
        "ip_6_e": 0,
        "ip_6_f": 0,
        "mintk": 273.15,
        "maxtk": 393.15,
        "minpbar": 1.01325,
        "maxpbar": 1.01325,
        "calcmode": "Entered",
        "dataquality": 1,
        "datasource": 6,
```

```
                                        "reference": "NotYetDetermined",
                                        "description": ""
                                }
                        }, ## Ende des Datenstzes zur ersten WW
                        { ## Hier beginnt der Datensatz für die zweite (hier
ternäre) WW
                                "Declaration":
                                {
                                        "interactiontype": "Pitzer_theta+psi",
                                        "pcon_1": "Na<+>",
                                        "pcon_2": "K<+>",
                                        "pcon_3": "Cl<->",
                                        "description": ""
                                },
                                "IP298":
                                {
                                        "ip298_1": -0.0120000000633,
                                        "ip298_1_unctype": "",
                                        "ip298_1_negativeunc": 0,
                                        "ip298_1_positiveunc": 0,
                                        "ip298_2": -0.001800002655,
                                        "ip298_2_unctype": "",
                                        "ip298_2_negativeunc": 0,
                                        "ip298_2_positiveunc": 0,
                                        "ip298_3": 0,
                                        "ip298_3_unctype": "",
                                        "ip298_3_negativeunc": 0,
                                        "ip298_3_positiveunc": 0,
                                        "ip298_4": 0,
                                        "ip298_4_unctype": "",
                                        "ip298_4_negativeunc": 0,
                                        "ip298_4_positiveunc": 0,
                                        "ip298_5": 0,
                                        "ip298_5_unctype": "",
                                        "ip298_5_negativeunc": 0,
                                        "ip298_5_positiveunc": 0,
                                        "ip298_6": 0,
                                        "ip298_6_unctype": "",
                                        "ip298_6_negativeunc": 0,
                                        "ip298_6_positiveunc": 0,
                                        "calcmode": "CTPFUNC",
                                        "dataquality": -1,
                                        "datasource": -1,
                                        "reference": "InternallyCalculated",
                                        "description": ""
                                },
                                "IPT":
                                {
                                        "tpfunc": "Pitzer-function",
                                        "ip_1_a": 0,
                                        "ip_1_b": -0.0120439272355523,
                                        "ip_1_c": 0,
                                        "ip_1_d": 0.000000147816722111973,
                                        "ip_1_e": 0,
                                        "ip_1_f": 0,
                                        "ip_2_a": 0,
                                        "ip_2_b": -0.0079019352865476,
                                        "ip_2_c": 0,
                                        "ip_2_d": 0.0000204660638523062,
                                        "ip_2_e": 0,
                                        "ip_2_f": 0,
                                        "ip_3_a": 0,
                                        "ip_3_b": 0,
                                        "ip_3_c": 0,
                                        "ip_3_d": 0,
                                        "ip_3_e": 0,
                                        "ip_3_f": 0,
                                        "ip_4_a": 0,
                                        "ip_4_b": 0,
                                        "ip_4_c": 0,
                                        "ip_4_d": 0,
```

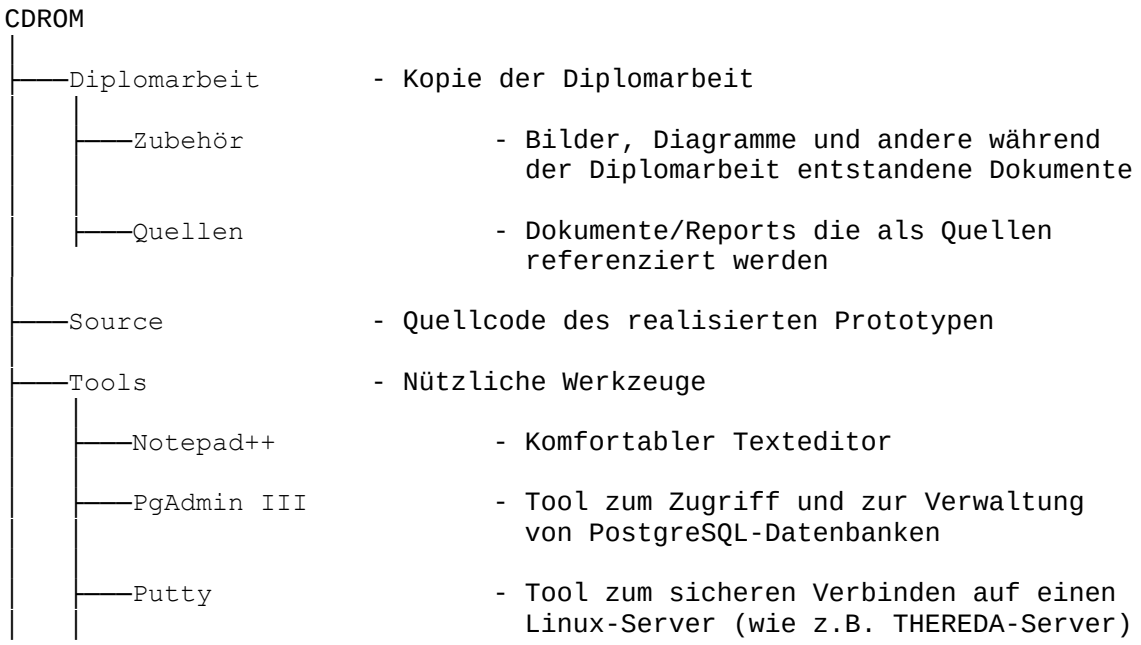
```
                                        "ip_4_e": 0,
                                        "ip_4_f": 0,
                                        "ip_5_a": 0,
                                        "ip_5_b": 0,
                                        "ip_5_c": 0,
                                        "ip_5_d": 0,
                                        "ip_5_e": 0,
                                        "ip_5_f": 0,
                                        "ip_6_a": 0,
                                        "ip_6_b": 0,
                                        "ip_6_c": 0,
                                        "ip_6_d": 0,
                                        "ip_6_e": 0,
                                        "ip_6_f": 0,
                                        "mintk": 273.15,
                                        "maxtk": 393.15,
                                        "minpbar": 1.01325,
                                        "maxpbar": 1.01325,
                                        "calcmode": "Entered",
                                        "dataquality": 1,
                                        "datasource": 6,
                                        "reference": "NotYetDetermined",
                                        "description": ""
                                }
                        } ## Ende des Datensatzes für zweite (ternäre) Pitzer-WW
                ], ## Ende der Liste aller Pitzer-WW
                ## Achtung: In einer JSON-Datei kann für die wässrige Phase nur
                ## entweder Pitzer oder SIT auftaucht. Lediglich zu illustrativen
                ## Zwecken snd in diesem Template beide Modelle simultan
vertreten.
                "interactionmodel": "SIT",
                "interaction":
                [ ## Hier beginnt die Liste aller SIT-WW
                        { ## Hier beginnt die erste SIT-WW
                                "Declaration":
                                {
                                        "interactiontype": "SIT_simple",
                                        "pcon_1": "UF3<+>",
                                        "pcon_2": "Cl<->",
                                        "pcon_3": "",
                                        "description": ""
                                },
                                "IP298":
                                {
                                        "ip298_1": 0.1,
                                        "ip298_1_unctype": "",
                                        "ip298_1_negativeunc": 0.1,
                                        "ip298_1_positiveunc": 0.1,
                                        "ip298_2": 0,
                                        "ip298_2_unctype": "",
                                        "ip298_2_negativeunc": 0,
                                        "ip298_2_positiveunc": 0,
                                        "ip298_3": 0,
                                        "ip298_3_unctype": "",
                                        "ip298_3_negativeunc": 0,
                                        "ip298_3_positiveunc": 0,
                                        "ip298_4": 0,
                                        "ip298_4_unctype": "",
                                        "ip298_4_negativeunc": 0,
                                        "ip298_4_positiveunc": 0,
                                        "ip298_5": 0,
                                        "ip298_5_unctype": "",
                                        "ip298_5_negativeunc": 0,
                                        "ip298_5_positiveunc": 0,
                                        "ip298_6": 0,
                                        "ip298_6_unctype": "",
                                        "ip298_6_negativeunc": 0,
                                        "ip298_6_positiveunc": 0,
                                        "calcmode": "Entered",
                                        "dataquality": 1,
                                        "datasource": 1,
```

```
                                        "reference": "GRE/FUG1992",
                                        "description": ""
                                },
                                "IPT":
                                {
                                        ## Bisher gibt es noch keine publizierten T-
Funktionen für SIT.
                                        ## Also bleibt das Objekt leer.
                                }
                        } ## Ende des Datensatzes zur ersten SIT-WW
                ] ## Ende der Liste aller SIT-WW
        }, ## Ende des Objektes "Interactions" für "aq"
        { ## Hier beginnt das Objekt der WW im Mischkristall "Olivine"
                "phase": "Olivine",
                "InteractionModel": "RKMP",
                "Interactions":
                [
                ## Gleiche Struktur wie zuvor
                ]
        } ## Ende des Objektes "Interactions" für Olivine
], ## Ende der Liste aller Interactions in allen Phasen

##########################################################################
# Jedes Kürzel zu Literaturangaben (references) in den obigen Datensätzen
# wird hier nun aufgeschlüsselt

"Bibliography": ## Viertes Objekt: Bibliographische Referenzen
[ ## Hier beginnt die Liste mit allen bibliographischen Referenzen
        { ## Hier beginnt die erste bibliographische Referenz
                "ID"                    : "HUM/AND2005",
                "type"          : "Book",
                "pubname"       : "",
                "origin"        : "OECD Nuclear Energy Agency (NEA)",
                "title"         : "Chemical Thermodynamics of compounds and
complexes of U. Np. Pu. Am. Tc. Se. Ni. and Zr with selected organic ligands",
                "author"        : "",
                "year"          : 2005,
                "volume"        : "",
                "page"          : "",
                "editors"       : "F. J. Mompean, M. Illemasshne, J. Perrone",
                "language"      : "English",
                "publisher"     : "Elsevier",
                "ISBN_ISSN"     : "0-444-51402-3"
        }, ## Ende der ersten bibliographische Referenz
        {
                "ID"                    : "ATK/GLA1992",
                "type"          : "Journal",
                "pubname"       : "Cement and Concrete Research",
                "origin"        : "",
                "title"         : "Cement Hydrate Phases: Solubility at 25° C.",
                "author"        : "Atkins, M., Glasser, F. P., Kindness, A.",
                "year"          : 1992,
                "volume"        : 22,
                "page"          : "241-246",
                "editors"       : "",
                "language"      : "English",
                "publisher"     : "",
                "ISBN_ISSN"     : ""
        },
        {
                "ID"                    : "PHI/HAL1988",
                "type"          : "Report",
                "pubname"       : "NUREG/CR-4864, LBL-22860, SAND87-0323",
                "origin"        : "",
                "title"         : "Thermodynamic tables for nuclear waste
isolation: Aqueous
                solutions database",
                "author"        : "Phillips, S. L., Hale, F. V.",
                "year"          : 1988,
                "volume"        : "",
                "page"          : "",
```

```
                        "editors"      : "",
                        "language"     : "English",
                        "publisher"    : "",
                        "ISBN_ISSN"    : ""
                },
                {
                        "ID"                    : "GIF1994",
                        "type"         : "Thesis",
                        "pubname"      : "",
                        "origin"       : "",
                        "title"        : "Influence des ions chlorure sur la chimie des
actinides",
                        "author"       : "Giffaut, E.",
                        "year"         : 1994,
                        "volume"       : "",
                        "page"         : "",
                        "editors"      : "",
                        "language"     : "French",
                        "publisher"    : "Universite de Paris-Sud, Orsay, France",
                        "ISBN_ISSN"    : ""
                }
        ] ## Ende der Liste aller bibliographischen Referenzen
} ## Ende des Objektes der gesamten THEREDA-DB
```

Anlage 2 - Inhalt CD-Rom

Auf der beigelegten CD-Rom befinden sich eine Kopie der Diplomarbeit, sowie alle während der Diplomarbeit entstandenen Dokumente. Weiterhin sind auf der CD der Quellcode des implementierten Prototyps und einige Dokumente aus der Quellenangabe.



Danksagung

Hiermit möchte ich mich bei allen Personen herzlich bedanken, die mich bei der Erstellung dieser Diplomarbeit unterstützt haben.

Für die Betreuung seitens der Hochschule für Technik und Wirtschaft Dresden bedanke ich mich bei Herrn Prof. Gunter Gräfe, der mich während meiner Diplomarbeit betreut und umfangreich unterstützt hat.

Besonderer Dank geht auch an meinen betrieblichen Betreuer Dr. Vinzenz Brendler vom Forschungszentrum Dresden-Rossendorf, der mir stets bei inhaltlichen und fachlichen Problemen zur Seite stand.

Abschließend danke ich noch meinen Eltern, die mir dieses Studium ermöglicht haben.

Eidesstattliche Erklärung

Hiermit versichere ich, Steffen Leske, geboren am 04.10.1982, dass die vorliegende Diplomarbeit von mir selbständig verfasst wurde. Zur Erstellung wurden von mir keine Weiteren als die angegebenen Hilfsmittel benutzt.

_____

Steffen Leske Dresden, 26. Januar 2010