Entwicklung einer Datenbasis für Anwendungen im Bauwesen

Dipl.-Ing. Irina Biltchouk, Technische Universität Berlin
Prof.Dr.Dr.h.c.mult. Peter Jan Pahl, Technische Universität Berlin

Zusammenfassung

Zur Speicherung von persistenten Objekten benannter Klassen einer Anwendung und unbenannter Klassen der Java - Plattform wurde eine dokumentübergreifende Datenbasis für Bauingenieuranwendungen entwickelt. In der Datenbasis wird die in Java übliche Klassendefinition ohne Zusatzinformation verwendet. Die Datenbasis ist auf einen Arbeitsspeicher und Dateien verteilt. Benannte Einzelobjekte werden mit ihrem Namen in die Datenbasis eingetragen, unbenannte Objekte erhalten einen Griff. Individuelle Objekte werden mit dem Namen oder Griff in der Datenbasis gelesen. Bei Bedarf werden Objekte aus Dateien nachgeladen.

1. Zielsetzung

Informationsmengen sind im Bauwesen traditionell mit Baudokumenten geordnet. Baudokumente werden in Computern überwiegend Datei - orientiert implementiert. Beispielsweise wird jeweils eine Zeichnung in einer Datei gespeichert. Zwischen den Dateien bestehen keine explizit gespeicherten Beziehungen. Die Informationsübertragung erfolgt vorwiegend mit den Dateien und ist folglich Dokument - gebunden. Das selektive Lesen oder Schreiben eines bestimmten Objektes in einer Zeichnung (beispielsweise zur Übertragung an einen Projektpartner) ist in der Regel nicht möglich. Der Ingenieur muss a priori wissen, auf welcher Zeichnung das Objekt dargestellt oder in welchem Bericht es berechnet ist. Zusätzlich muss er das Verzeichnis des Betriebssystems und die Datei kennen, welche die Zeichnung oder den Bericht enthält. Besondere Probleme treten immer dann auf, wenn die Informationen für ein Objekt über mehrere Zeichnungen und Berichte verteilt sind. Der Ingenieur ist häufig unsicher, ob er sich in allen relevanten Dokumenten informiert hat.

In Rahmen des DFG - Forschungsprojektes „Untersuchung von Strukturen in Informationsmengen des Bauwesens“ sollen die Vor- und Nachteile einer Dokument - freien Struktur der Information im Bauwesen bestimmt werden. Dazu werden die folgenden generalisierten Systemkomponenten entwickelt:

- Datenbasis für Objekte mit persistenten Identifikatoren
- Werkzeuge für den dynamischen Aufbau und Abbau von dynamischen Relationen zwischen den Objekten
- Werkzeuge für den dynamischen Aufbau und Abbau von Modellen
- Visualisierer für Darstellungen der Datenbasis
- Aktualisierer für geänderte Datenbasen
- Navigator für Datenbasen

In diesem Vortrag wird ein Konzept für die generalisierte Datenbasis einer Anwendung im Bauwesen vorgestellt.

2. Eigenschaften einer Datenbasis

Die Objekte einer Anwendung seien mit eindeutigen persistenten Identifikatoren versehen. Die Datenbasis dient zur persistenten Speicherung dieser Objekte. Eine Datenbasis, die bei der Ausführung von Methoden zweckmäßig genutzt werden kann, besitzt folgende Eigenschaften:

- Jede Sitzung einer Anwendung besitzt genau einen Arbeitsspeicher für die Datenbasis. Er enthält diejenigen Objekte, auf die Anweisungen von Methoden effizient mit dem Identifikator zugreifen können. Bei Bedarf sollen Objekte automatisch aus Dateien in den Arbeitsspeicher geladen werden.

- Der Anwender soll bestimmen können, in welcher Datei ein bestimmtes Objekt persistent gespeichert wird.

- Die Zugriffszeit auf ein Objekt in der Basis soll im Mittel von der Größenordnung der Zugriffszeit auf ein Objekt in einer Java - Methode sein.

- Die Datenbasis kann Objekte enthalten, die persistente Kollektionen von persistenten Objekten sind.

- Referenzen eines Objektes auf abhängige Objekte werden zu einem durch den Anwendungsentwickler gewählten Zeitpunkt gesetzt.

3. Eignung vorhandener Datenbanken und des Serialisierungskonzeptes von Java

Relationale und objektorientierte Datenbanken wurden hinsichtlich ihrer Eignung als Datenbasen für die vorliegende Aufgabe untersucht. Die genannten Datenbanktypen wurden mit dem Ziel entwickelt, große Mengen von Daten aufzubewaren und Informationen anhand von assoziativen Suchkriterien effizient aufzufinden. Das macht ihren Einsatz für entsprechende Aufgaben im Bauingenieurwesen zweckmäßig.

Die Analyse der Datenmodelle relationaler und objektorientierter Datenbanken führt zu der Erkenntnis, dass ihre mathematische Struktur gleich ist. Beide eignen sich zur Speicherung von Äquivalenzrelationen. Die zwei Datenbanktypen unterscheiden sich in ihrem Zweck. Der relationale Typ bietet vorrangig Anwenderoberflächen zur Eingabe, Suche und Darstellung von Informationen. Der objektorientierte Typ eignet sich vorrangig für das Zusammenwirken mit einem Programmiersystem. Jeder der beiden Typen kann um die Funktionalität des anderen Typs erweitert werden.

In einer relationalen Datenbank wird eine Relation konzeptionell auf eine Tabelle abgebildet, deren Eintragungen mit Schlüsseln identifiziert sind. Suchergebnisse werden in Tabellen dargestellt. Das rechnerinterne Speicherungsmodell ist produktabhängig. In einer objektorientierten Datenbank wird eine Relation auf eine Klasse abgebildet. Die Datenbank kann eigene Klassen besitzen oder die Klassen des Programmiersystems übernehmen sofern das Programmiersystem Strukturabfragen in Klassen (reflection) unterstützt. Jedes Objekt ist eine Prägung der Klasse, die temporäre und persistente Identifikatoren besitzt. Die Identifikatoren der Objekte werden in der Datenbasis verwaltet.

Für die vorliegende Aufgabe sind die beiden Datenbanktypen nur in beschränktem Maße geeignet. Benötigt wird eine Datenbasis mit möglichst effizientem Zugriff auf das Einzelobjekt, wenn sein Identifikator bekannt ist. Während einer Sitzung besitzen die beiden Datenbanktypen keinen Arbeitsspeicher im Arbeitsbereich der Anwendung, in dem Objekte der Datenbank wahlweise mit ihren temporären oder persistenten Identifikatoren ansprechbar sind und bei Bedarf automatisch aus den persistenten Dateien geladen werden. Die in anderem Zusammenhang zweckmäßige Such- und Präsentationsfunktionalität der Datenbanken wird für die vorliegende Aufgabe nicht benötigt und verursacht unnötigen Aufwand, der den Zugriff auf die Objekte im Vergleich zur Speicherung primitiver Datentypen in Dateien langsam macht.

Als Alternative für die Datenhaltung wurde auch das Konzept der Serialisierung von Objekten in Java untersucht. Der Schreib- und Lesemechanismus der Serialisierung besitzt folgende Eigenschaften, die eine Nutzung dieses Verfahrens für Bauingenieuraufgaben in vielen Fällen unzweckmäßig machen:

- Objekte werden in derselben Reihenfolge in die Datei geschrieben und aus der Datei gelesen. In Bauingenieuraufgaben unterscheidet sich die aufgabengerechte Reihenfolge des Lesens häufig von der aufgabengerechten Reihenfolge des Schreibens. Es ist auch üblich, häufig zwischen Schreiben und Lesen zu wechseln und dabei jeweils kleine Teilmengen der Datei zu übertragen. Der Bearbeitungszustand des Projektes bestimmt, welche Teilmenge der Daten zu übertragen ist.

- Ein Objekt wird mit allen von ihm abhängigen Objekten (seinem vollständigen Objektgraphen) in die Datei geschrieben und aus der Datei gelesen. Es ist nicht möglich, nur Teilmengen des Objektgraphen zu übertragen.

- Enthält der Objektgraph ein Objekt, das zuvor bereits in den ObjectOutputStream geschrieben wurde, so wird dieses nicht nochmals in den Stream geschrieben. Werden Attribute des Objektes zwischen beiden Schreibvorgängen geändert, so gehen diese Änderungen verloren und der Datenbestand wird inkonsistent.

Da die untersuchte Datenbanktypen und die Java Serialisierung die Anforderungen der vorliegenden Aufgabe nicht befriedigend erfüllen, wurde eine neue Datenbasis auf der Grundlage von Java entwickelt.

4. Konzept der Datenbasis

4.1 Persistenz von Objekten

Die Datenbasis einer Anwendung enthält die persistenten Objekte der Anwendung. Die Verwaltung der neuen Datenbasis unterscheidet zwischen benannten und unbenannten persistenten Objekten.

Ein Objekt heißt benannt, wenn ein String - Identifikator persistent in seinem Körper gespeichert wird. Der String - Identifikator des Objektes ist eindeutig im Identifikationsraum der Anwendung. Der persistente Identifikator heißt der Name des Objektes. Eine Klasse benannter Objekte heißt benannte Klasse. Eine benannte Klasse implementiert das Interface NamedObject. Dieses enthält eine Methode getName(), mit der die Namen der Objekte abgefragt werden. Die Namen aller benannten Objekte der Anwendung sind in der Datenbasis der Anwendung registriert. Benannte Klassen dürfen nicht das Interface Serializable implementieren.

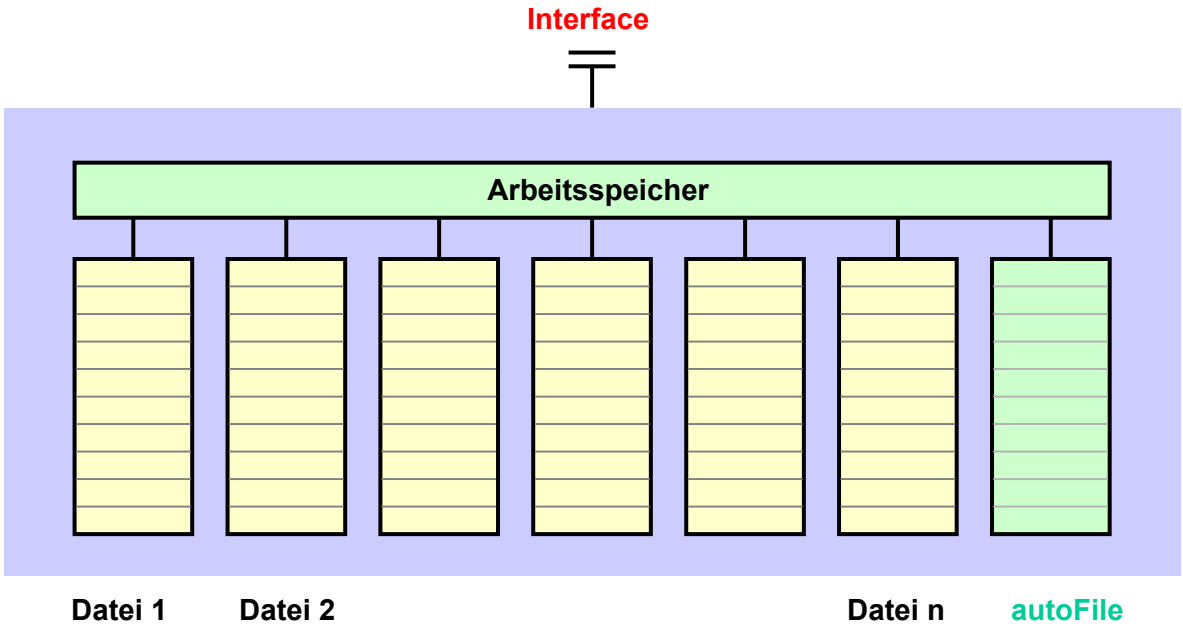
Ein Objekt einer Anwendung heißt unbenannt, weil es keinen Namen hat. Die Klassen der Pakete der Java - Plattform sind unbenannt. Die Datenbasis muss in der Lage sein, unbenannte Objekte zu speichern, weil Objekte solcher Klassen wie Color, Dimension, ArrayList und HashMap in bestimmten Anwendungen persistent sein müssen. Solche Objekte können für die Dauer ihrer Speicherung in der Datenbasis benannt werden. Dieser temporäre Name heißt der Griff des unbenannten Objektes. Der Griff wird dem Objekt automatisch von der Verwaltung der Datenbasis zugewiesen, wenn das Objekt zum ersten Mal in die Datenbasis gespeichert wird. Der Griff wird der Anwendung bekannt gemacht. Die Anwendung nutzt den Griff zum Adressieren, Lesen, Ändern oder Löschen des Objektes in nachfolgenden Operationen auf die Datenbasis. Unbenannte Objekte, die keine Kollektionen sind, werden in die Dateien der Datenbasis mit Hilfe von Java - Serialisation gespeichert. Ihre Klassen müssen deshalb das Interface Serializable implementieren. Kollektionen in Java

werden mit speziellen Methoden behandelt, weil sie benannte Objekte enthalten können und deshalb nicht serialisiert werden dürfen.

4.2 Speicherraum der Datenbasis

Während einer Sitzung ist die Datenbasis der Anwendung auf einen Arbeitsspeicher (ASP) und die geöffneten Dateien verteilt.

Bild 1 : Speicherraum der Datenbasis



Der Arbeitsspeicher enthält die Objekte, auf die die Methoden der Sitzung über Referenzen zugreifen können. Die temporäre Referenz eines Objektes im ASP heißt die Adresse des Objektes. Wenn das Objekt sich nicht im ASP befindet, ist seine Adresse null.

Zum Speichern und Lesen der Datenbasis werden Dateien mit direktem Zugriff der Klasse RandomAccessFile der Java - Plattform benutzt. Die Liste der Dateien, die Objekte für die aktuelle Sitzung enthalten, heißt die Dateiliste der Sitzung. Ein gegebenes Objekt der Datenbasis kann während einer Sitzung in mehreren Dateien oder in keiner Datei enthalten sein. Objekte der Sitzung können im ASP stehen, ohne in einer Datei enthalten zu sein.

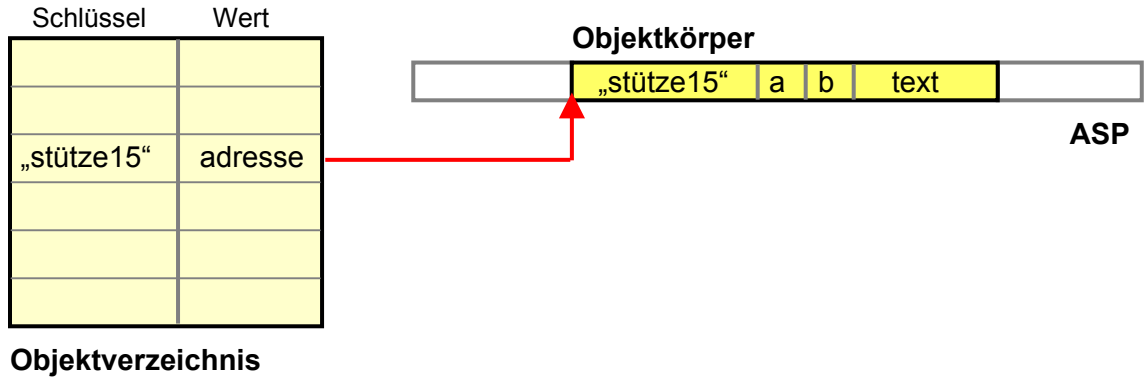
Wird ein Objekt in die Datenbasis eingetragen, so wird es automatisch in den ASP und in die Datei autoFile eingetragen. Wird autoFile auf null gesetzt, so wird das Objekt nur in den ASP eingetragen. Die Verwaltung der Datenbasis besitzt auch eine Methode, mit der ein Objekt in den ASP und in die Datei mit dem als Argument in dem Aufruf der Methode angegebenen Namen eingetragen wird. Die Information über das Objekt (Name oder Griff, Speicheradresse in der Datei, Speicherlänge und Typ) wird in der Tabelle dateiTabelle der angegeben Datei registriert.

Ein Körper eines Objektes im ASP heißt ein aktives Objekt. Ein Körper eines Objektes in einer Datei heißt ein passives Objekt. Besitzt ein Objekt Körper in mehr als einer Datei, so ist die Priorität jedes passiven Objektes gleich der Priorität seiner Datei. Ein Objekt aus der Datei mit dem Listenindex 0 besitzt die höchste Priorität unter den passiven Objekten. Das aktive Objekt besitzt die höchste Priorität. Die Eigenschaften des aktiven und eines passiven Objektes können verschieden sein.

4.3 Objektverwaltung in der Datenbasis

Wird ein benanntes Objekt während einer Sitzung in einer Methode konstruiert, so muss der Konstruktor dieses Objekt durch den Aufruf der Methode putObject(Object object) in der Basis registrieren. Die Datenbasis notiert das Paar (Name, Adresse) des neuen Objektes in einer HashMap objektVerzeichnis. Wird ein unbenanntes Objekt in die Datenbasis gespeichert, so weist die Verwaltung der Datenbasis ihm einen Griff (einen temporären Namen) zu. Das Paar (Griff, Adresse) wird im Objektverzeichnis registriert. Der Griff wird der Anwendung mitgeteilt.

Bild 2 : Objektverzeichnis im ASP



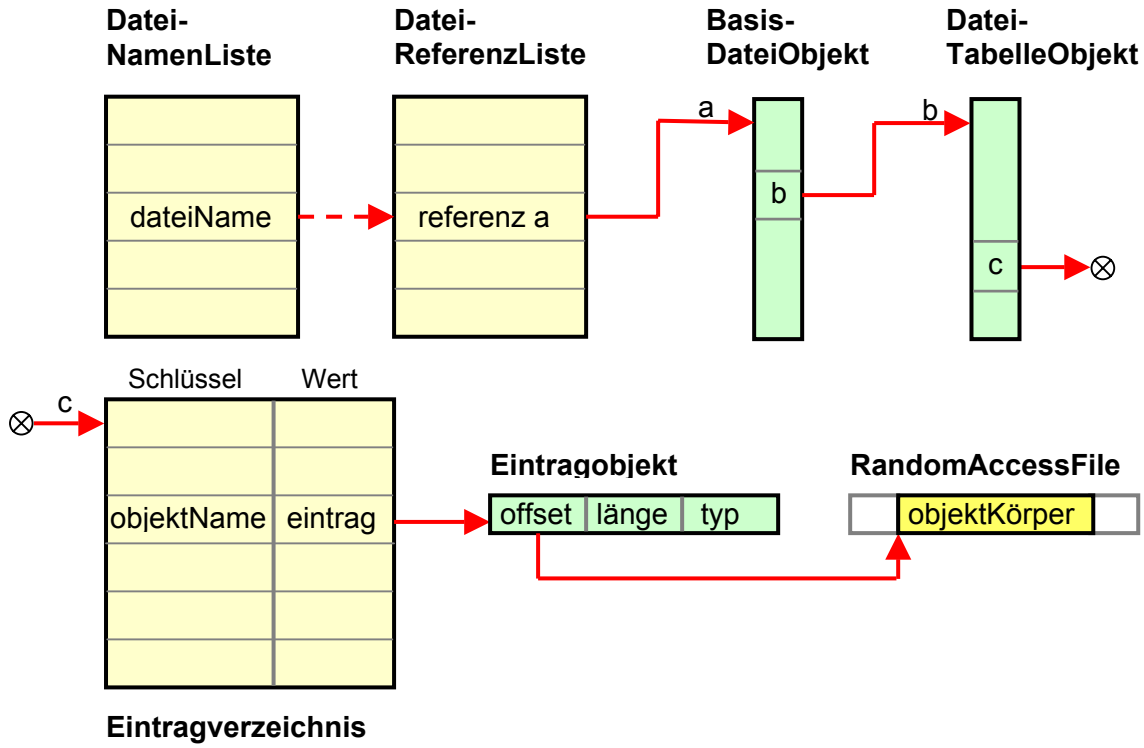
Wird die Verwaltung der Datenbasis während der Sitzung beauftragt, ein persistentes Objekt mit gegebenem Namen oder Griff aus einer Datei zu lesen, so wird das Paar (Name, Adresse) oder (Griff, Adresse) für dieses Objekt im Objektverzeichnis des ASP registriert, nachdem der Körper aus der Datei gelesen wurde.

Soll ein Objekt, das in der Datenbasis enthalten ist, von einer Methode einer Sitzung benutzt werden, so wird die Datenbasis mit einer get - Methode aufgerufen und liefert die Adresse des aktiven Objektes. Der get - Methode werden der Name oder der Griff des Objektes und wahlweise der Name einer Datei als Argument übergegeben.

Wird die Datenbasis nur mit dem Namen oder dem Griff des Objektes aufgerufen, so liefert sie die Adresse des Objektes mit der höchsten Priorität. Gibt es kein aktives Objekt mit dem gegebenen Namen oder Griff, so wird das passive Objekt mit der höchsten Priorität aus seiner Datei in den ASP gelesen. Die get - Methode liefert die Adresse des Objektes im ASP.

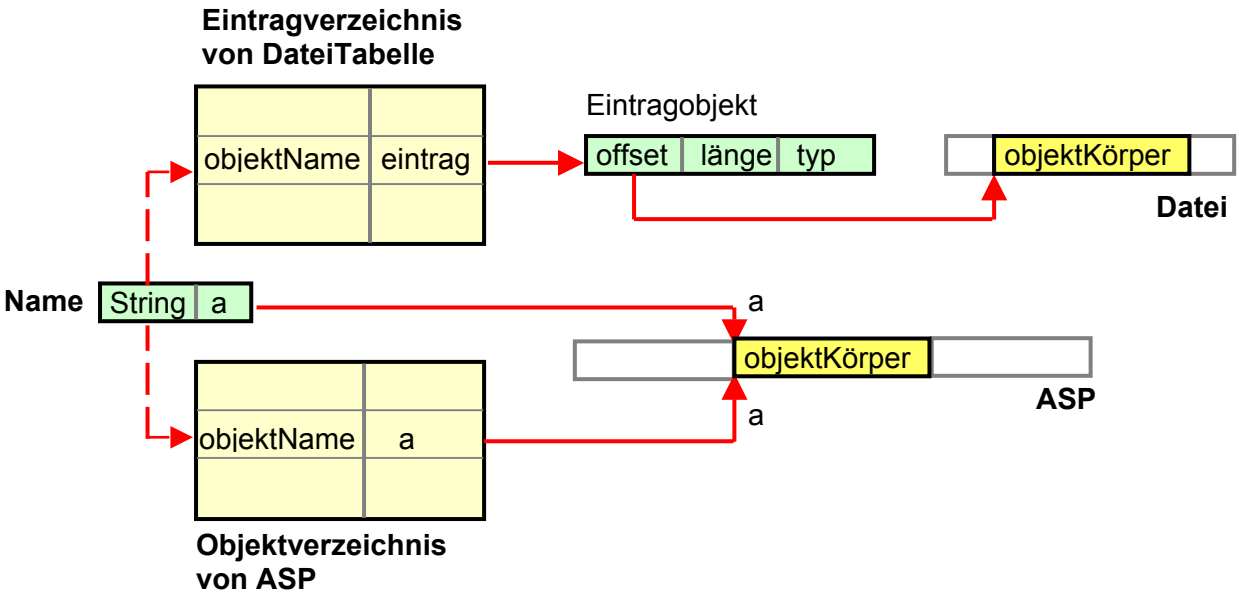
Zum Lesen eines Objektes in einer Datei wird die Datenbasis mit dem Namen oder Griff des Objektes und dem Namen der Datei aufgerufen. Existiert im ASP ein aktives Objekt mit dem angegebenen Namen oder Griff, so wird es in dem ASP gelöscht. Das Objekt wird danach aus der angegebenen Datei in den ASP gelesen. Seine Adresse wird in der Basis notiert. Die get - Methode liefert die Adresse des Objektes im ASP oder den Wert null, wenn die Datei das Objekt nicht enthält.

Bild 3 : Zugriff zu einem Objekt in einer Datei
Methode getObjectInFile(String objektName, String dateiName)



Persistente Objekte besitzen zwei Identifikatoren - die Adresse im ASP und den Namen mit Datentyp String. Diese Identifikatoren werden in einem Objekt der Klasse Name zusammengefasst. Referenzen von Name - Objekte sind in benannten Objekten, Kollektionen und Arrays enthalten. Wird für ein Objekt die Methode setReferences(Object objekt) aufgerufen, so werden mit Reflect die Referenzen seiner Name - Objekte erkannt. Die in den Name - Objekten gespeicherte Strings werden dazu benutzt, die temporären Adressen in der Datenbasis abzufragen und in das Name - Objekt einzutragen.

Bild 4 : Name - Objekt



Die Datenbasis besitzt zusätzlich Methoden für folgende Zwecke :

- Feststellen, ob ein Objekt in der Datenbasis vorhanden ist.
- Ein Objekt aus dem ASP, einer Datei oder der Datenbasis zu entfernen.
- Ein Objekt in eine Datei zu übertragen oder zu kopieren.
- Alle Objekte einer angegebenen Datei vom ASP in eine Datei zu schreiben oder aus einer Datei in den ASP zu lesen.
- Alle Objekte in einer angegebenen Datei oder in allen Dateien zu löschen.

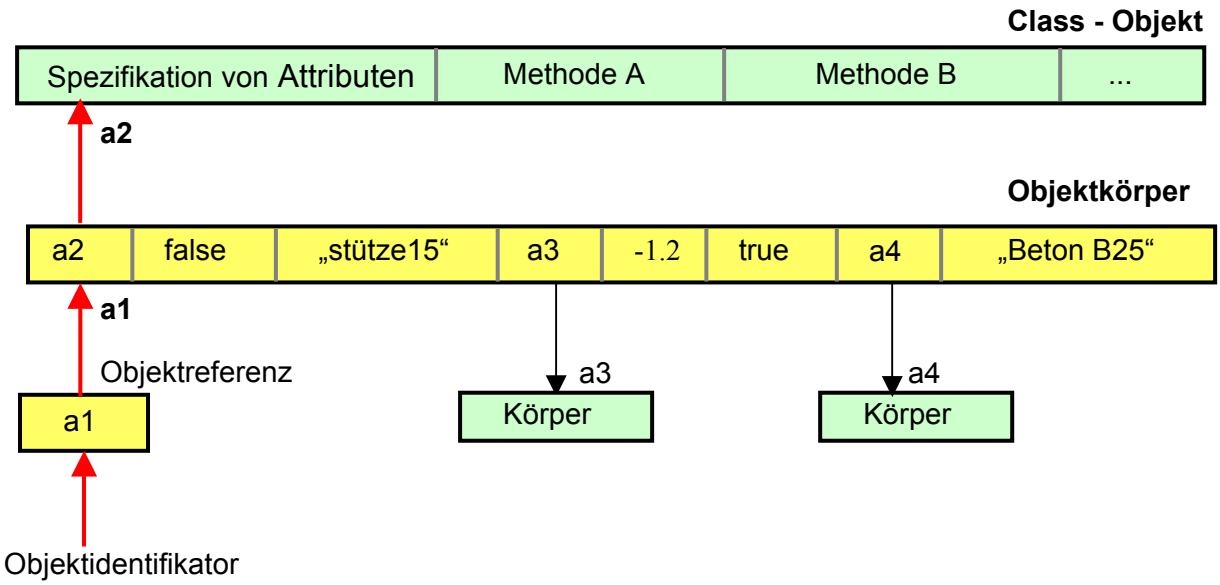
4.4 Speicherung unabhängiger Objekte in der Datenbasis

Die Verwaltung der Datenbasis unterscheidet zwischen unabhängigen und abhängigen Objekten. Ein Objekt heißt unabhängig, wenn der Entwickler die Verwaltung der Datenbasis beauftragt, dieses Objekt in die Datenbasis zu schreiben oder dort zu lesen. Ein Objekt B heißt von einem Objekt A direkt abhängig, wenn ein Attribut von A einen Identifikator des Objektes B enthält. Ein Objekt heißt abhängig, wenn es direkt oder indirekt von einem unabhängigen Objekt abhängt. Das gleiche Objekt kann unabhängig in einer Operation der Datenbasis und abhängig in einer anderen sein.

Gibt der Anwender die Anweisung, ein benanntes Objekt in eine Datei zu schreiben, so wird nur dieses Objekt in die Datei geschrieben. Im Gegensatz zur Serialisierung in Java werden abhängige benannte Objekte nicht in die Datei geschrieben. Abhängige unbenannte Objekte werden gespeichert. Das gespeicherte Objekt ersetzt eine eventuell bereits in der Datei vorhandene Kopie desselben Objektes. Wird das Objekt aus der Datei gelesen, so ersetzt es die aktive Version des Objektes im ASP. Will der Entwickler mehr als eine Kopie eines benannten Objektes speichern, so muss er dazu verschiedene Dateien verwenden.

Für das Zerlegen von Objekten in primitive Datentypen beim Schreiben benannter Objekte und bei ihrer Rekonstruktion beim Lesen wird das Java Konzept Reflection (Spiegelung) verwendet. Die unbenannten Objekte werden serialisiert. Das Schreiben und Lesen von Kollektionen als unbenannten Objekten wird in Abschnitt 4.6 erläutert.

Bild 5 : Struktur eines Objektes



Die Klassen des Application Programmer Interface (API) von Java, die zur Abfrage der Klassen- und Objektstruktur eingesetzt werden, sind in dem Paket java.lang.reflect zusammengefasst. Zur Abfrage der Klassenstruktur des zu speichernden benannten Objektes wird ein Objekt der Klasse Class aufgerufen, das den Name der Klasse des benannten Objektes liefert. Dieser wird in die Datei geschrieben und beim Lesen des benannten Objektes aus der Datei zum Aufruf des Class - Objektes verwendet. Mit der Objektmethode newInstance() der Klasse Class wird das benannte Objekt rekonstruiert. Für die Rekonstruktion des benannten Objektes beim Lesen aus einer Datei muss die benannte Klasse einen leeren Konstruktor besitzen. Der Aufruf der Methode getFields() für das Class - Objekt liefert ein Feld mit den Objekten der Klasse Field. Die Eigenschaften der Attribute des benannten Objektes werden durch Abfrage der Field - Objekte bestimmt. Die Klasse Modifier wird zur Abfrage der Java - Modifikatoren von Klassen und Attributen verwendet. Die transienten Attribute werden beim Speichern übergangen. Die Skalare oder Arrays Attribute werden in die Datei geschrieben, wenn ihr Datentyp primitiv, String oder Name ist. Ist der Datentyp des Attributes eine unbenannte Klasse, so wird das Objekt in die Datei serialisiert. Wird ein Objekt aus der Datei gelesen, so werden die transienten Attribute auf Ersatzwerte gesetzt, wenn ihr Datentyp primitiv oder String ist, und auf null, wenn ihr Datentyp Object ist. Skalare und Arrays werden aus der Datei gelesen, wenn ihr Datentyp primitiv, String oder Name ist. Ist ihr Datentyp eine unbenannte Klasse, so wird das Objekt aus der Datei deserialisiert.

Beim Speichern eines unbenannten unabhängigen Objektes der Datenbasis, dessen Klasse das Interface Serializable implementiert, wird mit der Anweisung writeObject(Object objekt) des Interface ObjectOutput die Folge seiner Attributwerte nach dem Java Konzept erstellt und in die Datei geschrieben. Beim Lesen aus der Datei wird das Objekt mit den in der Datei gespeicherten Werten rekonstruiert. Bei jedem Schreiben eines unabhängigen serialisierbaren Objektes in die Datei wird eine neue Kopie mit den aktuellen Eigenschaften des Objektes in die Datenbasis geschrieben. Eine eventuell vorhandene Kopie des Objektes wird durch die neue Kopie ersetzt.

4.5 Speicherung abhängiger Objekte in der Datenbasis

Abhängigkeit von Objekten: Im allgemeinen Fall ist ein Objekt A die Wurzel von mehreren abhängigen Objekten B, die ihrerseits die Wurzeln weiterer abhängiger Objekte C sein können. Ein abhängiges Objekt besitzt im allgemeinen Fall mehrere Wurzeln. Das Objekt A kann seinerseits ein abhängiges Objekt sein und eines oder mehrere Objekte aus B und C können seine Wurzeln sein.

Die Struktur der Objekte der Datenbasis ist im allgemeinen ein gerichteter Graph mit den Objekten als Knoten und den Identifikatoren als Kanten. Die Menge der Knoten wird in der Datenbasis gespeichert. Die Kanten können entweder in den Name - Objekten (Referenz, String) oder in den Tabellen der Verwaltung der Datenbasis (Dateizeiger) gespeichert sein.

Zu einem gegebenen Zeitpunkt befindet sich in der Regel nicht der vollständige Objektgraph einer Wurzel im ASP. Ein Teil der Knoten und Kanten des Objektgraphen sind deshalb in der Sitzung nicht zugänglich. Es ist die gemeinsame Aufgabe des Anwendungsentwicklers und der Verwaltung der Datenbasis, sicherzustellen, das der vorhandene Objektgraph für die vorliegende Aufgabe ausreichend ist. Dieser Objektgraph hängt von der Logik der gegebenen Aufgabe ab und muss vom Entwickler bestimmt werden. Für den Zugang zu den Daten und den Datentransport ist die Verwaltung der Datenbasis verantwortlich.

Implementierung : Enthält ein benanntes Objekt A ein abhängiges Objekt B, so ist das Objekt B entweder ein benanntes Objekt der Anwendung oder ein unbenanntes Objekt der Java - Plattform.

Enthält ein benanntes Objekt A ein unbenanntes Objekt B, so enthält der Körper von A die Referenz von B als Identifikator. Die Referenz von B wird in das Objekt A eingetragen wenn das Objekt B konstruiert wird.

Enthält ein benanntes Objekt A ein benanntes Objekt B, so enthält der Körper von A ein Objekt der Klasse Name mit dem Namen und der Referenz von B. Der Name des Objektes B kann in das Name - Objekt von A eingetragen werden, bevor das Objekt B konstruiert ist. Der Programmentwickler bestimmt den Zeitpunkt, zu dem die Referenz von B im Name - Objekt von A durch den Aufruf der Methode setReferences(Object objekt) der Datenbasis gesetzt wird.

Ein abhängiges unbenanntes Objekt wird mit dem Serialisierungverfahren in die Datei geschrieben. Die gespeicherte Version des abhängigen Objektes bleibt im folgenden auch dann unverändert, wenn die Kopie des Objektes im ASP geändert wird. Wird das Objekt zu einem späteren Zeitpunkt gelesen, so ist sein Wert gleich dem Wert des Objektes zu dem Zeitpunkt, zu dem es in die Datei geschrieben wurde.

Wird ein abhängiges benanntes Objekt in die Datei geschrieben, so wird nur der Name des Objektes in die Datei geschrieben. Beim Lesen aus der Datei wird ein Name - Objekt erzeugt und der Name des Objektes als Attribut darin gespeichert. Die Referenz des Objektes wird im Name - Objekt auf null gesetzt.

Abhängige benannte Objekte haben im Gegensatz zu abhängigen unbenannten Objekten die aktuelle Eigenschaften, die sich von den Eigenschaften unterscheiden können, die diese Objekte hatten, als das Wurzelobjekt in die Datei geschrieben wurde. Der Entwickler bestimmt den Inhalt der abhängigen benannten Objekte durch die Wahl des Zeitpunktes, zu dem diese Objekte in die Datei geschrieben und aus der Datei gelesen werden.

4.6 Speicherung der Kollektionen und Arrays

Kollektionen : Die Java - Plattform enthält Klassen, in deren Objekte andere Objekte gesammelt werden. Häufig benutzte Kollektionsklassen sind HashSet, ArrayList, LinkedList und HashMap. Dies sind unbenannte Klassen. Im Gegensatz zu den unbenannten Objekten, die keine Kollektionen sind, können Kollektionen abhängige benannte Objekte enthalten. Diese benannten Objekte können nicht durch Serialisierung gespeichert werden. Die Kollektionen können deswegen auch nicht serialisiert werden und müssen mit speziellen Methoden in der Datenbasis behandelt werden.

Die Kollektionen können null - Referenzen, unbenannte Objekte und benannte Objekte enthalten. Die Speicherung der Kollektionen basiert auf folgenden Prinzipien :

- In den Kollektionen HashSet, ArrayList oder LinkedList und als Wert der Eintragung in HashMap werden Name - Objekte mit den Namen der benannten Objekte oder den Griffen der unbenannten Objekte gespeichert. Will der Anwendungsentwickler ein solches Objekt nutzen, so muss er die Methode setReferences(Object objekt) der Verwaltung der Datenbasis aufrufen, die die Referenzen in den Name - Objekten setzt.

- Wird die Kollektion gespeichert, so wird der Name oder Griff jedes Objektes der Kollektion als String in der Datei gespeichert. Diese Vorgehensweise verhindert die Bildung von Zyklen, da ein String keine abhängigen Objekte enthält. Wird die Kollektion aus der Datei gelesen, so werden die Namen oder Griffe als Strings aus der Datei gelesen und in die Name - Objekte der Kollektion eingetragen.

- Ein Objekt A sei in einer Kollektion B enthalten. Wird B gespeichert, so wird A nicht automatisch ebenfalls gespeichert. Will der Entwickler das Objekt A in die Datei speichern, so muss er das Objekt A explizit als unabhängiges Objekt speichern. Zu diesem Zweck enthält die Datenbasis die Dienstmethoden writeCollection(Object objekt, String dateiName) und readCollection (Object objekt,String dateiName).

Wird die Datenbasis beauftragt, ein unbenanntes Objekt in eine Datei einzutragen, so verzweigt sie entsprechend der Klasse des Objektes. Ist die Klasse HashSet, ArrayList, LinkedList oder HashMap, so werden der Klassenname und die Größe der Kollektion als Skalare in die Datei geschrieben. Dann wird für jedes Objekt der Kollektion ein String in die Datei geschrieben. Dieses String ist der Name eines benannten Objektes, der Griff eines unbenannten Objektes oder das Schlüsselwort „_NULL_“ für null - Referenzen.

Arrays : Eine Menge primitiver Datentype oder Objekte wird Java - Array genannt, wenn jedes Element der Menge den gleichen Namen, den gleichen Datentyp und die gleiche Anzahl von Indizes besitzt. Die Elemente des Arrays werden eindeutig durch die Werte ihrer Indizes unterschieden.

Zur Speicherung der Struktur eines Arrays werden die Datentypen seiner Wurzel und seiner Komponenten sowie die Längen der Komponenten in die Datei gespeichert. Enthält eine Wertkomponente primitive Datentypen oder Strings, so werden diese sequentiell in die Datei geschrieben.

Enthält eine Wertkomponente die Objekte, die keine Strings sind, so kann jedes Objekt die Wurzel eines Objektgraphen der Datenbasis sein. Die Zyklenbildung wird analog zu Kollektionen verhindert. Ein Name - Objekt, das den Namen oder Griff des Objektes enthält, wird in den Array gespeichert. Es ist die Verantwortung des Anwendungsentwicklers, die entsprechende Objekte als unabhängige Objekte in die Datei zu speichern. Zu diesem Zweck enthält die Datenbasis die Dienstmethoden writeArray(Object array, String dateiName) und readArray(Object array, String dateiName).

4.7 Funktionalität der Datenbasis

Die Schnittstelle der beschriebenen Datenbasis besteht aus Methoden zur Verwaltung von Dateien und Methoden zur Verwaltung von unabhängigen Objekten. Die Vollständigkeit des aktuellen Satzes von Methoden wird zur Zeit in einer Pilotanwendung überprüft.

Methoden zur Verwaltung von Dateien :

```
boolean   openFile        (String fileName)
boolean   openFile        (String fileName, int priority)
boolean   closeFile       (String fileName)
boolean   clearFile       (String fileName)
boolean   containsFile    (String fileName)
boolean   setFilePriority (String fileName, int priority)
void      setAutoFile     (String autoFile)
String    getAutoFile()
void      closeAllFiles()
void      clearAllFiles()
ArrayList getFileList()
ArrayList getFileListForObject   (String objectName)
void      clearWSP()
```

Methoden zur Verwaltung von unabhängigen Objekten :

```
boolean   putObject         (Object object)
String    putObjectInBase   (Object object,String fileName)
```

```
Object      getObject           (String objectName)
Object      getObjectInBase     (String objectName,String fileName)
Object      copyObjectToFile    (String objectName,String fileName)
Object      moveObjectToFile    (String objectName,String fileName)
boolean     containsObject      (String objectName)
boolean     containsObjectInFile (String objectName,String fileName)
boolean     removeObject        (String objectName)
boolean     removeObjectInFile  (String objectName,String fileName)
boolean     removeObjectInAllFiles (String objectName)
boolean     moveObjectsFromWSPToFile (String fileName)
boolean     readObjectsFromFileToWSP (String fileName)
ArrayList   setReferences       (Object object)
void        removeReferences    (Object object)
void        writeCollection     (Object object,String fileName)
void        readCollection      (Object object,String fileName)
void        writeArray          (Object array,String fileName)
void        readArray           (Object array,String fileName)
```

5. Schlussfolgerungen

Die Entwicklung der beschriebenen Datenbasis hat zu mehreren wichtigen Konzepten geführt :

- Unterscheidung zwischen benannten und unbenannten persistenten Objekten einer Anwendung.
- Unterscheidung zwischen abhängigen und unabhängigen persistenten Objekten einer Datenbasis.
- Automatische Umsetzung von persistenten Identifikatoren (Strings) und temporären Identifikatoren (Referenzen) mit Hilfe der Klasse Name zur Speicherung benannter abhängiger Objekte, der Kollektionen und Arrays.
- Setzen der Referenzen auf benannte Objekte zu dem Zeitpunkt, zu dem sie von Methoden gebraucht werden.
- Transfer von Objekten durch ihre Referenzen.
- Beeinflussung der Dateibelegung durch den Anwendungsprogrammierer.

Die Programmierung der Datenbasis ist abgeschlossen und in sich geprüft. Die Untersuchung der Eignung der Datenbasis zur Entwicklung der anderen in Abschnitt 1 genannten Systemkomponenten steht noch aus. Es ist zu erwarten, dass dieser Erprobungsprozess zu Änderungen im vorliegenden Basiskonzept führen wird.

6. Förderung

Die Verfassern danken der Deutschen Forschungsgemeinschaft für die Förderung der vorliegenden Arbeiten im Rahmen des DFG - Forschungsvorhabens DFG-Gz. PA 162/9-1 („Untersuchung von Strukturen in Informationsmengen des Bauwesens“).

7. Literaturverzeichnis

Pahl, Peter Jan / Damrath, Rudolf
Mathematische Grundlagen der Ingenieurinformatik
Springler-Verlag Berlin Heidelberg New York 2000

Kemper, Alfons / Eickler, Andre
Datenbanksysteme
R.Oldenbourg Verlag München Wien 1996

Thalheim, Bernhard
Entity-Relationship Modeling. Foundation of Database Technology
Springler-Verlag Berlin Heidelberg New York 2000

Rechenberg, Peter / Pomberg, Gustav
Informatik - Handbuch
Carl Hanser Verlag München Wien 1997
Abschnitt E3
K. Dittrich
Datenbanksysteme

Schmidt, Duri
Persistente Objekte und objektorientierte Datenbanken
Konzepte, Architektur, Implementierung und Anwendung
Carl Hanser Verlag München Wien 1991