16 Client und Server

In diesem Kapitel wird zunächst die Aufteilung der Kernkomponenten auf Client und Server sowie der grundlegende Kommunikationsfluss zwischen diesen skizziert (Kap. 16.1). Daran anschließend werden zwei Strategien vorgestellt, die zur Verbesserung der Performance zwischen Server und Client eingesetzt werden (Kap. 16.2).

16.1 Aufteilung und Kommunikation

Die Kernkomponenten *Selektion* und *Auswertung* sind entsprechend des gewählten Entwurfs im Client Layer der Schnittstelle angesiedelt, also in dem Teil der Software, die auf dem Rechner des jeweiligen Anwenders abläuft. Die Kernkomponente *Abbildung* residiert hingegen im Server Layer und wird zentral als Dienst für alle individuellen Clients bereitgestellt. Abb. 16.1 veranschaulicht die drei Schichten-Architektur der Schnittstelle, ihre Aufteilung auf Client Layer, Server Layer und Data Layer sowie die Nutzung des Internet als Infrastruktur zur Kommunikation zwischen den Schichten.

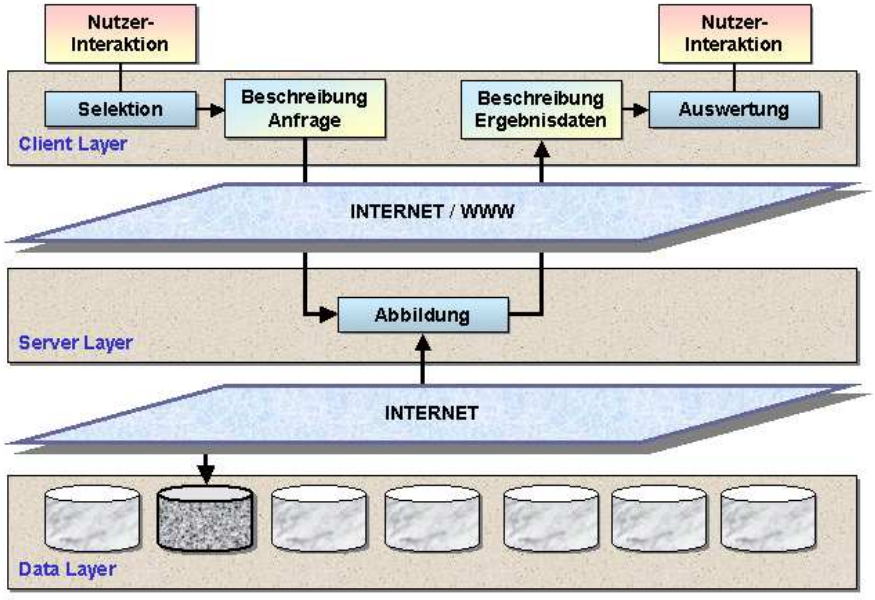

Abb. 16.1 - Die Drei-Schichten-Architektur der Schnittstelle mit Aufteilung der Kernkomponenten zwischen Client Layer und Server Layer.

Hat der Anwender über einen Client eine Anfrage formuliert, wird von diesem eine entsprechende Anfragebeschreibung generiert und über Internet vom Anwenderrechner an den Server der Schnittstelle weitergeleitet. Dieser analysiert die Anfragebeschreibung, identifiziert die auszuwählende Datenbank und greift, wiederum über Internet, auf den Server des jeweiligen RDBMS zu. Die Ergebnisdaten werden vom Server in die interne Ergebnisrepräsentation transformiert, wobei in der Anfragebeschreibung enthaltene Konfigurationsinformationen Verwendung finden. Die so entstandene Ergebnisrepräsentation wird anschließend vom Server wieder zurück an den aufrufenden Client übertragen und von diesem dem Anwender zur interaktiven Auswertung zugänglich gemacht.

16.2 Performance-Tuning

Nachfolgend werden zwei Strategien beschrieben, die zur Steigerung der Performance eingesetzt werden und im Rahmen ihrer iterativen Erweiterung in die Schnittstelle integriert wurden.

16.2.1 Unterstützung von Tabellenkaskaden

Die gewählte Vorgabe, einzelne Datenräume jeweils in Form einer Datenbanktabelle bereitzustellen (vgl. Kap. 13.6), kann im gegebenen Anwendungskontext in einigen Fällen zu sehr großen Datenbanktabellen führen, die durch Kreuzprodukte bei der Zusammenfassung mehrerer Ausgangstabellen entstehen. Da sich die Größe einer Datenbanktabelle in der Regel direkt auf die Geschwindigkeit auswirkt, mit der ein RDBMS Anfragen auf dieser ausführt, können hieraus unerwünschte Performance-Einbußen resultieren. Eine derartige Situation entstand aus der sich im Laufe des Betriebs ergebenden Anforderung, die Funktionalität der Schnittstelle um Möglichkeiten zur Selektion von Stationen anhand jahresgenauer statistischer messdatenbezogener Kriterien zu erweitern (vgl. Kap. 20.9). Hierfür ist die Einbeziehung von Zeitreihenmetadaten erforderlich, die neben Informationen über einzelne Stationen zusätzlich jahresgenaue Kennwerte aller von diesen erhobenen Variablen enthalten.

Das hieraus entstehende Anwachsen des einzubeziehenden Datenvolumens soll an einem Beispiel verdeutlicht werden. So benötigt beispielweise eine Datenbanktabelle, die pro Zeile eine Station anhand jeweils stationsspezifischer Attribute wie Identifikator, Name und Georeferenzierung dokumentiert, für die Beschreibung von 5.000 Stationen entsprechend 5.000 Zeilen. Wird diese Tabelle zusätzlich um ein Attribut erweitert, um einzelne Variablen zu unterscheiden und werden pro Station jeweils 10 Variablen dokumentiert, erhöht sich die Zahl der erforderlichen Zeilen um den Faktor 10, so dass nun bereits 50.000 Zeilen erforderlich sind. Werden nun weitere Attribute hinzugefügt, um für jede Variable die Werteausprägungen einzelner Messjahre zu dokumentieren und pro Variable für jedes von 70 Jahren entsprechende Informationen hinzugefügt, erhält man bereits eine Tabellengröße von $70 * 50.000$, also 3.500.000 Zeilen[266].

Um nun im Interesse der Performance je nach Kontext einer nutzerdefinierten Anfrage jeweils nur das jeweils erforderliche Datenvolumen einbeziehen zu müssen, können statt einer großen Datenbanktabelle drei Datenbanktabellen mit unterschiedlichem Informationsgehalt (stationsspezifische Informationen, stationsspezifische Informationen plus Informationen über einzelne Variablen sowie stationsspezifische Informationen plus jahresgenauen Informationen über einzelne Variablen) als individuelle Datenräume in die Schnittstelle eingebunden werden. Dabei ist das durch die redundante Vorhaltung resultierende Datenvolumen für die Bereitstellung dreier Tabellen (mit 5.000 + 50.000 + 3.500.000 = 3.555.000 Zeilen für obiges Beispiel) nur unwesentlich größer als das Datenvolumen, dass alleine für die Tabelle mit den jahresbezogenen Informationen benötigt wird. Der Nachteil dieses Ansatzes liegt vielmehr im erhöhten kognitiven Aufwand, den er dem Anwender abverlangt, der auf diese Weise statt mit nur einem mit drei verschiedenen Datenräumen konfrontiert wird.

Eine geeignetere Lösung (vgl. Abb. 16.2) besteht darin, dem Anwender in solchen Fällen den Zugriff auf einen einzigen Datenraum zu suggerieren und Anfragen automatisch und transparent auf die jeweils kleinste zu ihrer Beantwortung geeignete Datenbanktabelle abzubilden. Auf diese Weise muss der Anwender nicht zwischen mehreren Datenräumen auswählen; zugleich können diejenigen Anfragen, die nur einen Bruchteil des Gesamtvolumens einbeziehen müssen, deutlich performanter realisiert werden. Eine entsprechende Abbildungsfunktionalität wurde in den Server integriert und erfolgt transparent für den Client. Der Setupmechanismus des Servers erlaubt für jeden eingebundenen Datenraum die Konfiguration einer Kaskade aus mehreren Datenbanktabellen. Eine ankommende An-

[266] Bei einer Erweiterung bspw. auf monats- oder tagesgenaue Informationen wächst das Datenvolumen entsprechend weiter an.

frage gegen einen solchen Datenraum wird anhand der in ihr verwendeten Attribute ausgewertet, die kleinste zu ihrer Beantwortung geeignete Tabelle identifiziert und für die Durchführung der Anfrage ausgewählt.

Abb. 16.2 - Transparenter Zugriff auf eine Tabellenkaskade durch automatische Abbildung einer Anfrage.

16.2.2 Dynamische Abbildung von Ergebniswerten

Typische Anfrageergebnisse enthalten häufig Zeichenketten wie „*meteorology*" oder „*min_temperature*", die mehrfach in verschiedenen Datensätzen auftreten. Die wiederholte Darstellung solcher Bezeichner ist unumgänglich, um eine intuitive Ergebnisinterpretation zu ermöglichen; ihr mehrfacher Transport vom Server zum Client erhöht jedoch das zu übertragende Datenvolumen und damit den Zeitaufwand bis zur Bereitstellung angeforderter Ergebnisdaten. Eine Verringerung des erforderlichen Datendurchsatzes ohne Informationsverlust für den Anwender kann dadurch erreicht werden, dass anstelle der längeren Zeichenketten für die jeweiligen Werteausprägungen kurze, eindeutige Identifikatoren zum Client übertragen und dort für die Darstellung automatisch in die interpretierbaren Bezeichner umgesetzt werden. Um eine entsprechende Reduktion des zu übertragenden Datenvolumens zu unterstützen, enthält der Client der Schnittstelle einen bidirektionalen Mechanismus, der für individuelle Datenräume und dort für individuelle Datenbankattribute konfiguriert werden kann und es erlaubt, Datenbankwerte in Darstellungswerte und Darstellungswerte wieder zurück in Datenbankwerte abzubilden[267]. Die Abbildung der Werte wird für jedes Attribut über eine eigene Abbildungstabelle (Look-Up-Table, LUT) gesteuert, die vom Client bei Bedarf zur Laufzeit eingelesen wird. Die interne Verwaltung jeweils einer Look-Up-Table und die entsprechenden Wertetransformationen übernehmen hierfür entworfene Komponenten, die als DynamicValueMapper (DVM) bezeichnet werden (vgl. Abb. 16.3).

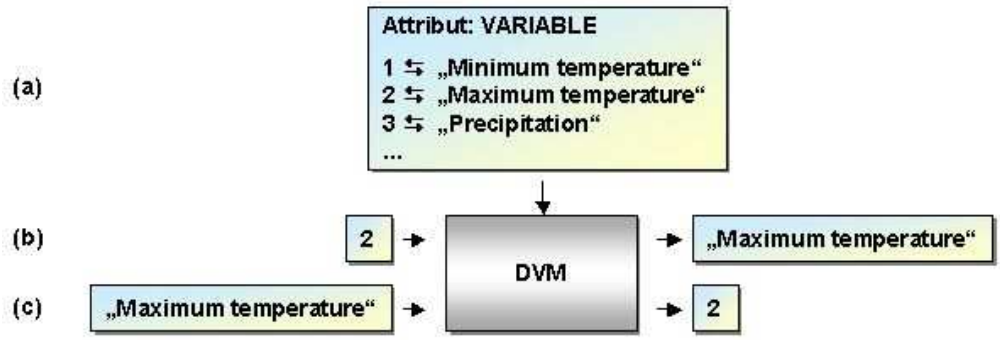

Abb. 16.3 - Schematische Darstellung der Funktionsweise einer DynamicValueMapper-Komponente: (a) Konfiguration; (b) Abbildung von Datenbank- auf Darstellungswerte; (c) Abbildung von Darstellungs- auf Datenbankwerte.

[267] Die Abbildung von Darstellungswerten in Datenbankwerte ist für die Anfragegenerierung erforderlich.

Die für einen Datenraum jeweils vorzunehmenden Transformationen werden den Filter- und Auswertungsmodulen transparent zur Verfügung gestellt. Diese Aufgabe übernimmt der DvmManager, der sich entsprechend des vom Anwender ausgewählten Datenraumes konfiguriert und die erforderlichen DynamicValueMapper generiert. Immer dann, wenn ein Übergang zwischen Darstellungs- und Datenbankwelt zu erfolgen hat, genügt es, einen Wert zusammen mit dem entsprechenden Attributnamen an den DvmManager zu übergeben, um automatisch eine eventuell erforderliche Transformation vornehmen zu lassen.

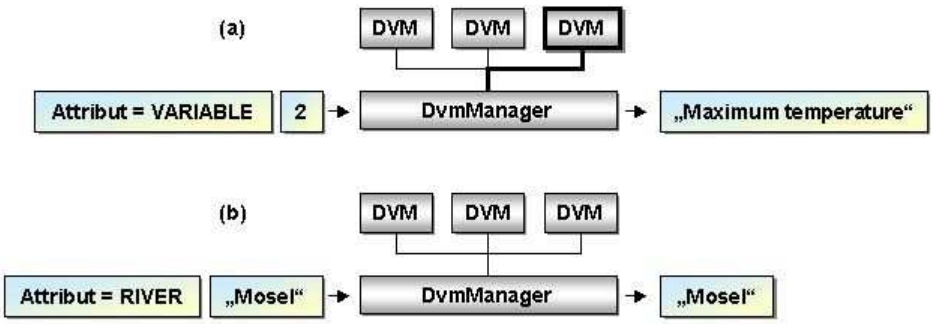

Abb. 16.4 - Schematische Darstellung der Funktionsweise des DvmManager: (a) Identifikation und Verwendung der geeigneten DVM-Komponente für eine Wertetransformation; (b) unveränderte Rückgabe von Werten, für die keine Transformation konfiguriert wurde.

Der DvmManager analysiert, ob für das übergebene Attribut eine Abbildungsvorschrift definiert wurde; ist dies der Fall, identifiziert er den hierfür zuständigen DynamicValueMapper und kommuniziert mit diesem, um eine geeignete Werteabbildung vornehmen zu lassen (vgl. Abb. 16.4a). Ist hingegen für das gegebene Attribut keine Abbildungsvorschrift konfiguriert, wird der hereingereichte Wert vom DvmManager unverändert zurückgegeben (vgl. Abb. 16.4b).