Johann Wolfgang Goethe Universität
Frankfurt am Main

Proseminar SS 2000

Dynamic Queries
Steuerelemente

Fabian Wleklinski
(fabian@wleklinski.de)

13. August 2000

1 Inhaltsverzeichnis

2 Vorwort

Das Aussterben nicht-visueller Anwendungen für den gemeinen Anwender ist allgegenwärtig, und macht auch vor der Datenbankabfrage nicht halt. Die technologischen Fortschritte der neunziger Jahre in Bezug auf Massenspeicher und Netzwerke haben zu einem Overkill an Daten geführt, der mit herkömmlichen Datenbankabfragen nur noch schwer zu bewältigen ist.

Dynamic Queries versprechen eine Lösung für dieses Problem: kryptische SQL-Ausdrücke und Kommandozeilen-Werkzeuge weichen der Echtzeit-Simulation in der virtuellen Realität.

Eine Datenbankabfrage wird nicht mehr gestartet, und läuft bis zu ihrem „Ende“, sondern ist vielmehr eine kontinuierliche Interaktion zwischen dem Anwender und der Anwendung. Im Sinne des sogenannten „*Tight Coupling*“ agieren und reagieren die Anwendung und der Anwender gegenseitig auf ihre Aktionen. Eine ausgefeilte Oberfläche vermittelt dem Anwender dabei das Gefühl, „live“ an der Abfrage teilzunehmen.

Die Anwendung greift der eigentlichen Datenbankabfrage voraus, indem Steuerelemente ihren Wertebereich so anpassen; dass keine unsinnigen Abfragen möglich sind. Umgekehrt kann das Abfrageergebnis erneut als Abfrage verwendet werden. Durch Mausklicks werden Abfragen kontinuierlich präzisiert und neu formuliert, der Abfragevorgang bekommt einen „browserartigen“ Charakter.

Dynamic Queries bedeuten neue Anforderungen an grafische Ressourcen, Algorithmen, Datenbankstrukturen, Schnittstellen und Protokolle, sowie an die verwendeten Steuerelemente. Dieser letztgenannte Aspekt ist der Inhalt dieses Papiers:

Die Herausforderungen für Steuerelemente durch *Dynamic Queries*.

3 Ein Beispiel zur Motivation: *Dynamic HomeFinder*

3.1 Einleitung

Ben Shneiderman unterrichtet seit mehr als 16 Jahren an der *University of Maryland*, und gehört zu den Pionieren der *Dynamic Queries*. Anfang der neunziger Jahre entwickelte sein Schüler Christopher Williamson die Anwendung „*Dynamic HomeFinder*“, um die Wirkung von *Dynamic Queries* durch „*Tight Coupling*“ auf Anwender zu demonstrieren, und zu erforschen.

Im Jahre 1994 gründet Christopher Williamson zusammen mit seiner Frau Cynthia Williamson die Firma „DreamQuest Software“ (http://www.dqsoft.com), die Computerspiele entwickelt und vertreibt.

Im Jahre 1995 entwickelt Christopher Williamson zusammen mit Tom Smallwood eine erweiterte Version des *Dynamic HomeFinder*, als Abschlussarbeit für den Titel „Masters of Engineering in Software Engineering“[1]. Diese Software kann unter der URL http://www.dqsoft.com/homefind/ abgerufen werden, und dient im Folgenden dazu, den Bedarf für neuartige Steuerelemente durch *Dynamic Queries* zu demonstrieren.

Der *Dynamic HomeFinder* ist eine Software, die die Suche von Immobilien aus einem vorgegebenen Datenbestand erlaubt, und sich dabei der Mittel von *Dynamic Queries*, insbesondere *Tight Coupling*, bedient.

3.2 Steuerelemente statt SQL

Wer schon einmal mit einem konventionellen Datenbanksystem gearbeitet hat, der kennt sie: die SQL-Befehle. Kryptische Kommandos in der Form wie „`select * from homes`“ veranlassen einen Datenbankserver, hunderte oder tausende Zeilen voller Werte auszudrucken.

Wesentlich eleganter geschieht die Datenbankabfrage mit dem *Dynamic HomeFinder*. Nach dem Programmstart präsentiert sich diese Anwendung wie folgt:

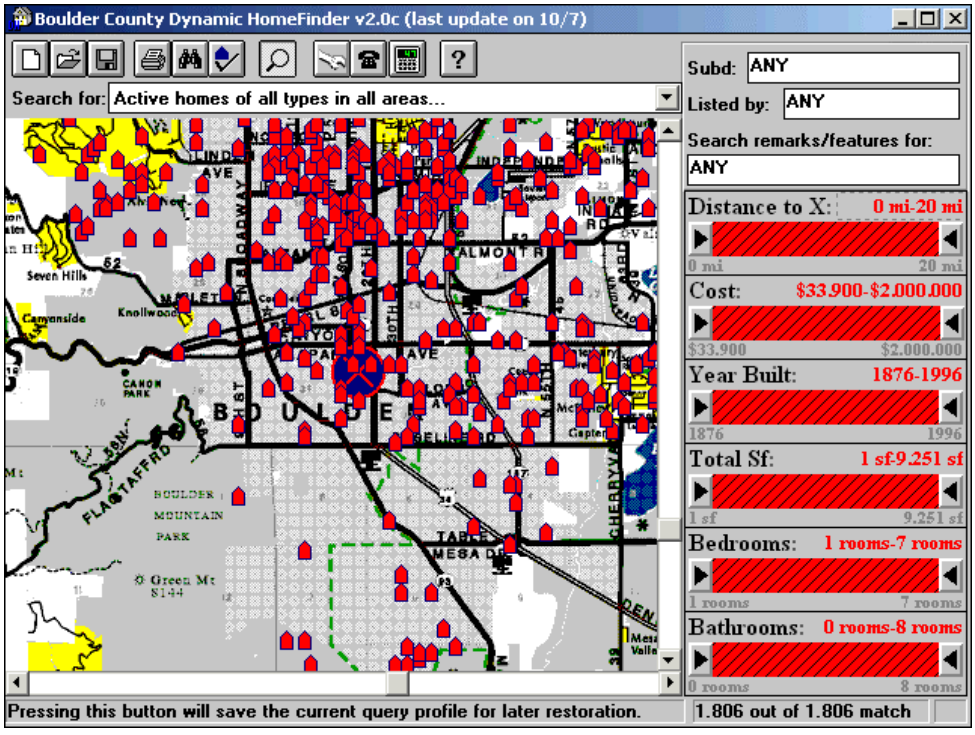

Abbildung 1 – *Dynamic HomeFinder* – SELECT * FROM HOMES

[1] Dieser Abschluss entspricht ungefähr dem deutschen Diplom

Im linken Teil des Bildschirmfotos ist ein „*Starfield Display*“ zu sehen, auf welchem jede gefundene Immobilie, d.h. jeder Datensatz, den die Anfrage beschreibt, durch ein kleines Symbol in Form eines Hauses dargestellt ist.

Im rechten Teil des Bildschirmfotos befinden sich die sogenannten „Steuerelemente“. Das sind die Bestandteile der Oberfläche einer Anwendung, die der Anwender (im Allgemeinen) mit der Maus betätigt (klickt, verschiebt, drückt, ...) und dadurch eine bestimmte Eingabe macht. Siehe auch „4 Steuerelemente“.

Durch einfaches und intuitives „Herumdrehen“ und „Ziehen“ an den Steuerelementen können die Auswahlkriterien für die Datenbankabfrage definiert werden, z.B. Ober- und Untergrenzen für „Anzahl der Badezimmer“, „Erstellungsjahr“ und „Preis“ einer Immobilie.

Während der Anwender durch das Betätigen der Steuerelemente im rechten Teil des Bildes seine individuellen Präferenzen definiert, wird das Abfrageergebnis in Gestalt der Ansicht im linken Teil des Bilder in Echtzeit aktualisiert.

Nachdem der Anwender seine Präferenzen definiert hat, ist die Menge der gefundenen Immobilien (Datensätze) geringer als zum Programmstart. Die Oberfläche sieht dann z.B. wie folgt aus:

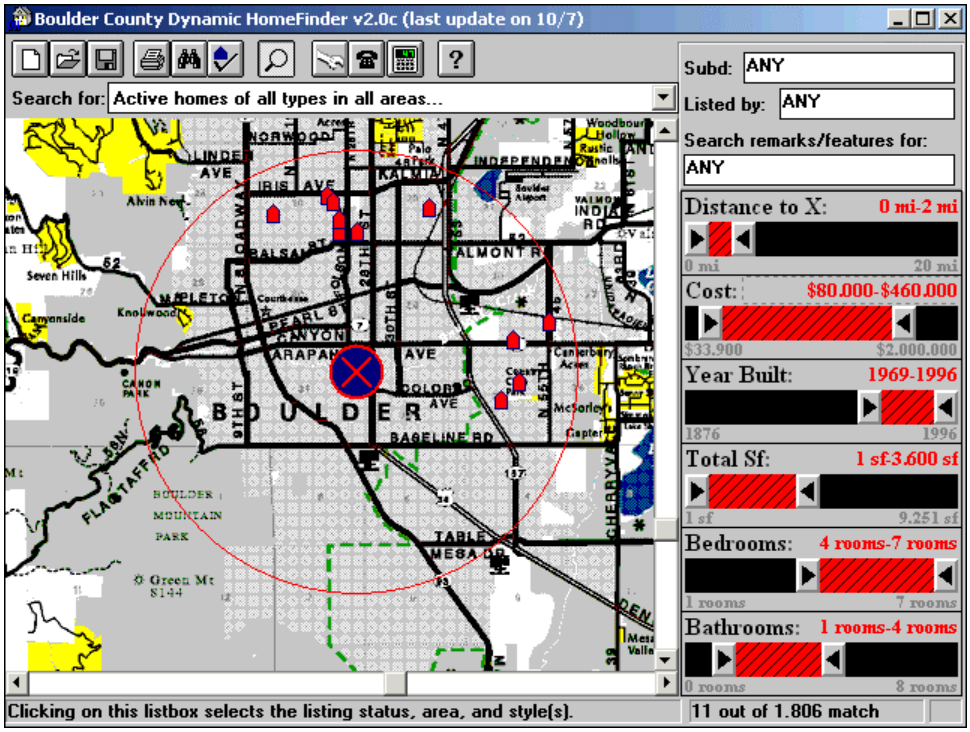

Abbildung 2 – *Dynamic HomeFinder* während einer typischen Abfrage

Eine solche Abfrage artet ohne die Benutzung von *Dynamic Queries* schnell in einem zeilenlangen und unübersichtlichen Konstrukt wie dem folgenden aus:

```
select * from homes where (cost>=80000) and (cost<=460000) and
(year>=1969) and (year<=1996) and (bedrooms>=4) and (bedrooms<=7)
and (bathrooms>=1) and (bathrooms<=4) and (distance<=2) and (to-
tal<=3600);
```

3.3 Übergang zur Problemstellung

Damit ist nun noch einmal am Beispiel des *Dynamic HomeFinder* gezeigt worden, was *Dynamic Queries* eigentlich sind, und wie man sie anwenden kann. Die Untersuchungen von Ben Shneiderman haben gezeigt, dass diese Art von Benutzeroberfläche von der Mehrheit der Anwender akzeptiert, ja sogar begrüßt wird. Ein Anwender soll sogar gesagt haben: „Ich will nicht aufhören, das macht Spaß!“

Ohne darauf einzugehen, sind beim *Dynamic HomeFinder* bereits erweiterte Steuerelemente zum Einsatz gekommen. Die verwendeten Schieberegler zur Auswahl von Ober- *und* Untergrenze werden nämlich von keiner populären, grafischen Oberfläche wie *MS Windows*, *KDE* oder *Swing* zur Verfügung gestellt, und sind von den allerwenigsten Anwendern schon einmal benutzt worden. Dass aber trotzdem jeder Anwender auf Anhieb mit der Anwendung arbeiten konnte, und sogar Spaß dabei empfunden hat, verdeutlicht die Wichtigkeit der Entwicklung neuer, und „guter“ Steuerelemente.

4 Steuerelemente

Nachdem in dem vorangegangenen Kapitel festgestellt wurde, was man mit Steuerelementen machen kann, und wie nützlich diese sind, geht es nun darum, Kriterien zu definieren, anhand derer man verschiedene Steuerelemente miteinander vergleichen kann.

Diese Kriterien werden in den folgenden Kapiteln benutzt, um die „Nachteile“ der klassischen, und die „Vorteile“ der neuartigen Steuerelemente herauszustellen.

4.1 Aufgaben der Steuerelemente

Steuerelemente haben im Wesentlichen drei Aufgaben:

1. Definieren einer Auswahl

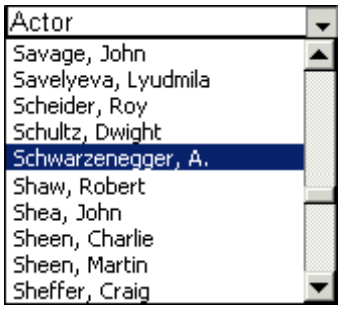

Abbildung 3 - Auswahlaufgabe

Der Anwender definiert eine Auswahl, um der Anwendung etwas „mitzuteilen“.

Dazu gehören z.B. das „Ziehen“ an einem Schieberegler, das „Anklicken“ eines Kontrollkästchens oder das Eingeben von Text in ein Texteingabefeld.

Hierbei reicht der Anwender Informationen an die Anwendung weiter.

2. Visualisierung von Informationen

Abbildung 4 - Informationsaufgabe

Nachdem der Anwender eine Auswahl getroffen hat (oder auch schon davor), wird diese Auswahl weiterhin angezeigt. Wenn beispielsweise an einem Schieberegler „gezogen“ wurde, verharrt der Schieberegler in der neuen Position, die der Anwender jederzeit ablesen kann.

Alleine das Vorhandensein eines Steuerelementes informiert den Anwender bereits, und zwar über die Existenz dieser Eingabemöglichkeit, und ggf. über den zulässigen Wertebereich, aus dem der Anwender auswählen kann.[2]

Hierbei reicht die Anwendung Informationen an den Anwender weiter.

3. Auslösen von Aktionen

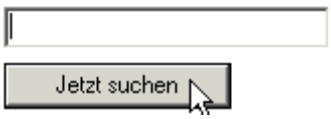

Abbildung 5 - Informationsaufgabe

Die Anwendung reagiert mit der Ausführung von Programmcode auf das Betätigen eines Steuerelementes. Dazu gehören typischerweise das „Klicken“ auf eine Schaltfläche, einen sogenannten „Button“.

Im Sinne des „*Tight Coupling*“ ist dieser Aspekt eines Steuerelementes für *Dynamic Queries* von untergeordneter Bedeutung, da Aktionen der Anwendung in Echtzeit geschehen, und nicht explizit durch Schaltflächen ausgelöst werden müssen.

[2] Bei dem Wertebereich handelt es sich z.B. im Fall eines Kontrollkästchens um „ja“ und „nein“, im Fall eines Texteingabefeldes um alle Texte, usw.

4.2 Kriterien für Steuerelemente

Kriterien für Steuerelemente lassen sich finden wie Sand am Meer. Die für *Dynamic Queries* relevanten lassen sich aber im Wesentlichen auf eine Handvoll reduzieren:

1. Platzbedarf

   Jedes Steuerelement benötigt Platz. „Platz“ im Sinne eines Bereiches auf dem Bildschirm, in dem das Steuerelement dargestellt wird, und der die dahinterliegenden Bereiche (in dem sich z.B. weitere Steuerelemente befinden können) verdeckt. Da die Größe des Bildschirms begrenzt ist, gilt es, mit der Ressource „Platz“ umsichtig zu wirtschaften.

   Insbesondere für *Dynamic Queries* ist der Platzbedarf eines Steuerelementes interessant, da es hierbei um die Abbildung einer Menge von Datensätzen auf einen Wertebereich geht. Stehen beispielsweise eine Million verschiedener Datensätze zur Verfügung, so wäre es wenig zweckmäßig, diese Datensätze alle untereinander in einer großen Liste darstellen zu wollen.

2. Diskrete Werte

   Je nach Verwendungszweck der Steuerelementes kann es gewünscht sein, dass diskrete Werte ausgewählt werden können – oder aber auch nicht. Das „perfekte Steuerelement“ würde natürlich beide Varianten zulassen, und die Auswahl dem Anwender überlassen.

   Da das „perfekte Steuerelement“ aber im Moment außer Reichweite ist, sollte ein Steuerelement vor allem die Auswahl diskreter Werte ermöglichen. Zwar mag z.B. bei der Auswahl von Größen wie „Lautstärke“ oder „Kaufpreis“ eine Toleranz von x Prozent sinnvoll sein, bei der Auswahl von „Anzahl der Badezimmer“ aber wohl kaum.

   Beispiel: Angabe der Anzahl der Badezimmer der gesuchten Immobilie.

3. Bereichsauswahl

   Ein Steuerelement sollte die Möglichkeit anbieten, sofern das im jeweiligen Kontext sinnvoll ist, nicht nur einen einzelnen Wert, sondern einen Bereich auszuwählen. Die Möglichkeit, einen einzelnen Wert auszuwählen, sollte natürlich erhalten bleiben.

   Beispiel: Angabe eines Wunschkaufpreises der gesuchten Immobilie mit Unter- und Obergrenze.

4. Mehrfachselektion

   Ein Steuerelement sollte die Möglichkeit anbieten, nicht nur einen einzelnen Wert, oder einen Wertebereich, sondern auch mehrere Werte oder Wertebereiche auszuwählen.

5. Browsen

   Für *Dynamic Queries* ist der Charakter des „browsen“ unabdingbar. Unter „browsen“ versteht man eine Art von zielgerichtetem Suchen, z.B. das Variieren eines Schiebereglers.

6. Informationsvisualisierung

   Wie in „4.1 Aufgaben der Steuerelemente“ dargelegt, haben Steuerelemente einen Informationszweck. Verschiedene Steuerelemente unterscheiden sich in dem Umfang und der Qualität, indem sie dem Anwender Informationen zur Verfügung stellen.

7. Bedienungskomfort

   Generell sollte ein Steuerelement für den Anwender so einfach wie möglich zu bedienen sein. Niedrigerer Bedienkomfort kann den Anwender entmotivieren, verärgern, und ggf. von der Benutzung und/oder dem Kauf abhalten.

Nachdem nun eine Handvoll Kriterien für die Bewertung von Steuerelementen gefunden ist, gilt es im folgenden Kapitel, die klassischen Steuerelemente anhand dieser Kriterienliste zu bewerten.

5 Die Problemstellung: klassische Steuerelemente

5.1 Einleitung

Jeder kennt Sie, die Schaltflächen, Schieberegler, Kombinationsfelder und Kontrollkästchen – oder eingedeutscht: „Buttons“, „Slider“, „Comboboxen“ und „Checkboxen“. Sie werden von Benutzeroberflächen und Betriebssystemen aller Art zur Verfügung gestellt, als kleine Auswahl seien hier *CDE*, *KDE*, *MS Windows*, *Mac OS*, *Swing*, *Open Look*, *Palm OS* genannt.

Es wird nun gezeigt, dass die klassischen Steuerelemente - trotz ihrer Allgegenwärtigkeit – keineswegs allen Kriterien gerecht werden, die *Dynamic Queries* an sie stellen.

5.2 Der Slider (Schieberegler)

Der Slider, oder: Schieberegler ermöglicht das Auswählen eines diskreten Wertes aus einem fest vorgegebenen Wertebereich. Ein klassisches Beispiel für die Verwendung des Schiebereglers ist der Einsatz als Lautstärkeregler:

Abbildung 6 – Ein Schieberegler

In Abbildung 6 sind sogar gleich zwei Schieberegler zu sehen, je einer für Lautstärke und Balance. Der Schieberegler besitzt den Vorteil, dass er mit einem konstanten Platz auskommt.

Der Schieberegler besitzt „theoretisch“ den Vorteil, dass diskrete Werte gewählt werden können. „Theoretisch“ deswegen, weil ab einer bestimmten Größe des Wertebereiches aus technischen Gründen[3] nur noch jeder x-te Wert gewählt werden kann. Im Falle des Lautstärkereglers ist das sicherlich nicht nachteilig, aber das sieht anders aus, wenn es sich z.B. um den Kaufpreis einer Immobilie handelt.

Das „browsen“ ist mit dem Schieberegler sehr gut möglich, und der Bedienkomfort ist relativ hoch.

Der klassische Schieberegler bietet aber keine Bereichsauswahl und auch keine Mehrfachselektion an. Die Informationsvisualisierung des Schiebereglers ist eher schlecht, denn er teilt dem Anwender nichts über den Wertebereich, den aktuellen Wert, die Werteverteilung oder dergleichen mit, und das, obwohl der Schieberegler über einige „freie Flächen“ verfügt, auf denen Informationen untergebracht werden könnten.

[3] Sowohl Maus als auch Bildschirm verfügen über eine bestimmte Auflösungsgrenze, innerhalb derer die Maus keine Bewegung mehr wahrnimmt, bzw. der Bildschirm keine Veränderung mehr darstellen kann.

Daher:

	Platz	Diskrete Werte	Bereich-Auswahl	Mehrfach-Selektion	„browsen“	Visualisierung	Komfort
Slider	+				+		+

5.3 Die Checkbox (Kontrollkästchen)

Das Kontrollkästchen oder die „Checkbox“ ermöglicht es, eine festdefinierte Option ein- oder auszuschalten. Es tritt meistens in Gruppierung mit anderen Kontrollkästchen auf.

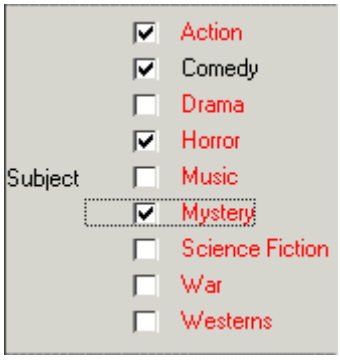

Abbildung 7 - Einige Kontrollkästchen

Der Platzbedarf einer Kontrollkästchen-Gruppe steigt linear mit der Größe des Wertebereiches – was es für eine Verwendung im Bereich der *Dynamic Queries* nur bedingt einsetzbar macht. Mehr als etwa zwanzig, oder auch dreißig verschiedene Optionen sind mit Kontrollkästchen (auch bei „großem“ Bildschirm) nicht visualisierbar. Der Einsatz eines Schieberegler, um zu erreichen, dass immer nur einige Kontrollkästchen gleichzeitig auf dem Bildschirm zu sehen sind, geht erheblich zu Lasten des Komforts.

Eine Gruppe von Kontrollkästchen erlaubt unabhängig von der Größe des Wertebereiches die Selektion und Mehrfachselektion diskreter Werte. Eine Bereichsauswahl ist durch die Mehrfachselektion zwar möglich, aber bei großem Wertebereich sehr unkomfortabel.

Von einem „browsen“ kann hier allerdings nicht die Rede sein. Wenn der Anwender alle x Kontrollkästchen gemäß seinen Vorstellungen eingestellt hat, und dann abrupt seine Meinung ändert, sind für ihn bis zu x manuelle Arbeitsschritte nötig. Bei einem Datenbestand von 1 Mio. Datensätzen erfordert das vom Anwender also bis zu 1 Mio. Mausklicks.

Die Informationsvisualisierung ist ausreichend, aber nicht perfekt. Dadurch, dass jede Option separat dargestellt wird, kann der Anwender sehen, welche Optionen zur Verfügung stehen. Es werden ihm aber keine darüber hinausgehenden Informationen angeboten, z.B. muss er eine Konfiguration von Kontrollkästchen erst ausprobieren, um festzustellen, was sich dahinter verbirgt.

Der Bedienungskomfort ist gut, verschlechtert sich aber zunehmend mit der Anzahl der Kontrollkästchen.

Daher:

	Platz	Diskrete Werte	Bereich-Auswahl	Mehrfach-Selektion	„browsen“	Visualisierung	Komfort
Checkbox		+	+	+			+

5.4 Der Radiobutton (Optionsfeld)

Das Optionsfeld, meistens „Radiobutton“ genannt, entspricht in wesentlichen Punkten der Checkbox, siehe „5.3 Die Checkbox (Kontrollkästchen)“:

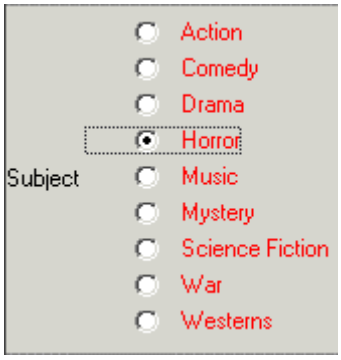

Abbildung 8 - Ein Optionsfeld

Anders als bei dem Kontrollkästchen kann immer nur einer, und nicht beliebig viele Werte ausgewählt werden. Typischerweise wird das Optionsfeld auch aus diesem Grund verwendet: um auszuschließen, dass der Anwender versehentlich mehrere Werte auswählt, obwohl eigentlich nur ein Wert sinnvoll ist.

Das Optionsfeld erfüllt die Kriterien also in gleichem Maße wie das Kontrollkästchen, mit der Ausnahme, dass Mehrfachselektion und Bereichsauswahl nicht möglich sind.

Daher:

	Platz	Diskrete Werte	Bereich-Auswahl	Mehrfach-Selektion	„browsen“	Visualisierung	Komfort
Radiobutton		+					+

5.5 Die Combobox (Kombinationsfeld)

Das Kombinationsfeld ermöglicht es, einen freien Text einzugeben, oder einen Text aus einer Liste von vorgegebenen Texten auszuwählen.

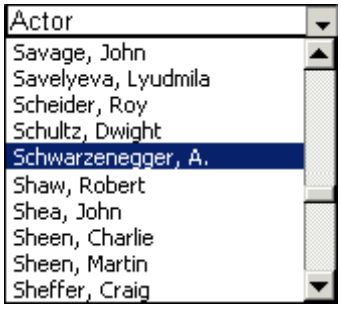

Abbildung 9 - Ein Kombinationsfeld

Der Platzbedarf des Kombinationsfeldes ist konstant. Die Liste, aus welcher der Anwender einen vorgegebenen Text auswählt, besitzt eine Konstante Hohe, in Abbildung 9 z.B. zehn Zeilen. Existieren in mehr vorgegebene Werte, als die Liste fasst, wird ein zusätzlicher Schieberegler angezeigt, wie in Abbildung 9 zu sehen ist.

Das Kombinationsfeld erlaubt es unabhängig von der Größe des Wertebereiches, einen diskreten Wert auszuwählen. Mehrfachselektion und Bereichsauswahl sind nicht möglich.

Von einem browsen kann hier ebenfalls nicht die Rede sein: Der Aufwand für den Anwender, ein Element in der Liste zu „finden“, und anzuwählen, steigt proportional zur Größe des Wertebereiches an. Daran ändert auch die Tatsache nichts, dass die Liste lexikographisch sortiert ist, und in einigen Implementierungen sogar ein schnelles Navigieren in der Liste mit den Buchstabentasten möglich ist[4].

Der Bedienungskomfort des Kombinationsfeldes ist relativ niedrig. Gleiches gilt für die Güte der Informationsvisualisierung.

Daher:

	Platz	Diskrete Werte	Bereich-Auswahl	Mehrfach-Selektion	„browsen“	Visualisierung	Komfort
Combobox	+	+					

5.6 Zusammenfassung

Zusammenfassend ergibt sich für die klassischen Steuerelemente das folgende Bild:

	Platz	Diskrete Werte	Bereich-Auswahl	Mehrfach-Selektion	„browsen“	Visualisierung	Komfort
Slider	+				+		+
Checkbox		+	+	+			+
Radiobutton		+					+
Combobox	+	+					

Summa summarum: Es gibt viel zu tun.

[4] Ein Druck auf „z“ bewirkt dann, dass der erste Eintrag der Liste, der mit „z“ beginnt, ausgewählt wird. Dadurch wird der Suchaufwand in etwa auf 1/26 reduziert.

6 Die Lösung (1): neuartige Steuerelemente

Es ist in den vorangehenden Kapiteln gezeigt worden, dass Steuerelemente für *Dynamic Queries* unabdingbar sind, und dass die klassischen Steuerelemente nicht allen Anforderungen gewachsen sind, die *Dynamic Queries* an sie stellen. Was wäre naheliegender, als zu versuchen, die klassischen Steuerelemente zu erweitern?

Im Folgenden werden Ansätze von Ben Shneiderman , Christopher Ahlberg und Stephen G. Eick gezeigt, durch die Erfindung neuartiger Steuerelemente die Unzulänglichkeiten der klassischen Steuerelemente auszuräumen, und so den Effekt des *Tight Coupling* zu verbessern, oder ihn erst zu ermöglichen.

6.1 Das Steuerelement “Alphaslider“

Der „Alphaslider“ ist eine von Ben Shneiderman und Christopher Ahlberg entwickelte Erweiterung des klassischen Schiebereglers[5]. Im Gegensatz zu dem klassischen Schieberegler besitzt der Alphaslider einen textuellen Wertebereich, und erlaubt optional die Auswahl eines Bereiches anstelle eines diskreten Wertes:

Abbildung 10 – Alphaslider

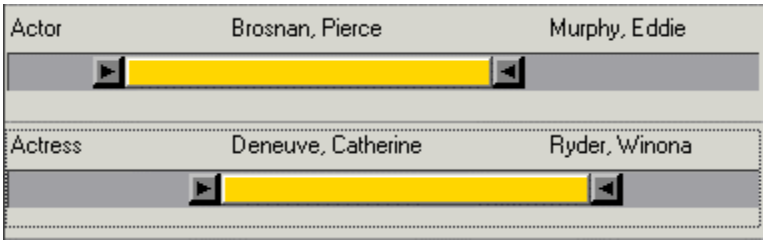

Abbildung 11 - Alphaslider (Bereichsmarkierung)

[5] Implementierungen findet sich u.a. in Ahlbergs Entwicklungen „*Dynamic HomeFinder*“ und „*Dynamic FilmFinder*“. Die verwendeten Bildschirmfotos entstammen dem „*Dynamic FilmFinder*“:

Je nach Sinn und Zweck kann der Entwickler einer Anwendung entweder fest den Modus zum Auswählen von Werten oder von Bereichen vorgeben, oder diese Wahl dem Anwender überlassen. Ben Shneiderman gibt in seiner Software „Dynamic FilmFinder“ dem Anwender über ein Kontextmenü die Möglichkeit, sich selber für einen Modus zu entscheiden:

Abbildung 12 - Alphaslider Kontextmenü 1

Abbildung 13 - Alphaslider Kontextmenü 2

Der Alphaslider von Ben Shneiderman verfügt zusätzlich über fortgeschrittene Visualisierungsfunktionen. In Abbildung 14 und Abbildung 15 ist zu sehen, wie sich das Aussehen des Alphasliders in Abhängigkeit der Existenz von Datensätzen für die jeweilige Einstellung verändert[6]:

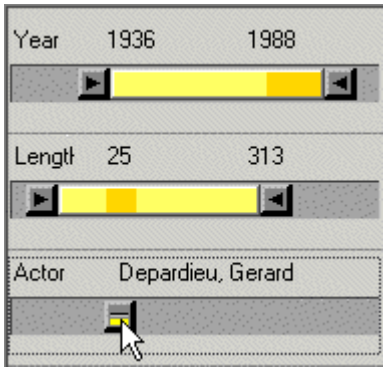

Abbildung 14 - Alphaslider Visualisierung 1

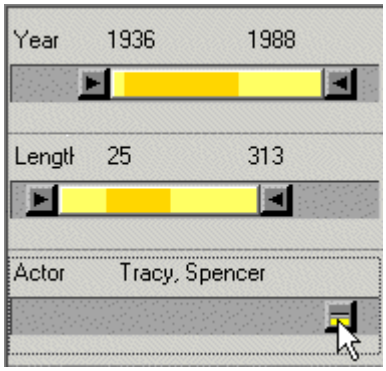

Abbildung 15 - Alphaslider Visualisierung 2

Der orangefarben bzw. dunkel gefärbte Bereich innerhalb des Alphasliders in Abbildung 14 und Abbildung 15 informiert den Anwender noch vor der Betätigung dieses Steuerelementes, dass es keine Filmdatensätze mit „Gerard Depardieu“ als Darsteller und einem Datum kleiner als etwa 1980 gibt. Betätigt der Anwender den Alphaslider zur Auswahl des Darstellers auf „Spencer Tracy“, bekommt er ohne Zeitverzug die Information, dass für diesen Darsteller keine Filmdatensätze mit einem Datum größer als etwa 1970 und kleiner als etwa 1940 existieren. Der Anwender kann dadurch in gleicher

[6] Obwohl es sich bei den entsprechenden Werten nicht um textuelle, sondern um numerische Wertebereiche handelt, sind hier Alphaslider verwendet worden

Zeit und mit gleichem Aufwand mehr Informationen aufnehmen, und wird außerdem davor geschützt, eine ungünstige oder gar unsinnige Auswahl zu treffen.

Der Alphaslider stimmt in wesentlichen Punkten mit dem klassischen Schieberegler überein, siehe „5.2 Der Slider (Schieberegler)“. Der Alphaslider benötigt ebenfalls konstanten Platz, und erlaubt ebenfalls weder Mehrfachselektion, noch die Auswahl bestimmter, diskreter Werte (einen entsprechend großen Wertebereich vorausgesetzt).

Die herausragende Eigenschaft des Alphasliders ist die Möglichkeit, nun auch in textuellen Wertebereichen zu browsen. Zur Auswahl textueller Werte dient unter den klassischen Steuerelemente nur das Texteingabefeld, die Combobox, die Checkbox und der Radiobutton – und mit all diesen Steuerelemente ist bislang kein browsen möglich.

Daher:

	Platz	Diskrete Werte	Bereich-Auswahl	Mehrfach-Selektion	„browsen“	Visualisierung	Komfort
Alphaslider	+		+		+	+	+

6.2 Das Steuerelement “Data Visualization Slider”

Der „Data Visualization Slider“ ist eine von Stephen G. Eick entwickelte Verbesserung des klassischen Schiebereglers. Das hervorstechendste Merkmal dieses Steuerelementes ist es, dass es über insgesamt vier Betriebsmodi verfügt, zwischen den der Entwickler bzw. der Anwender je nach Bedarf wählen kann:

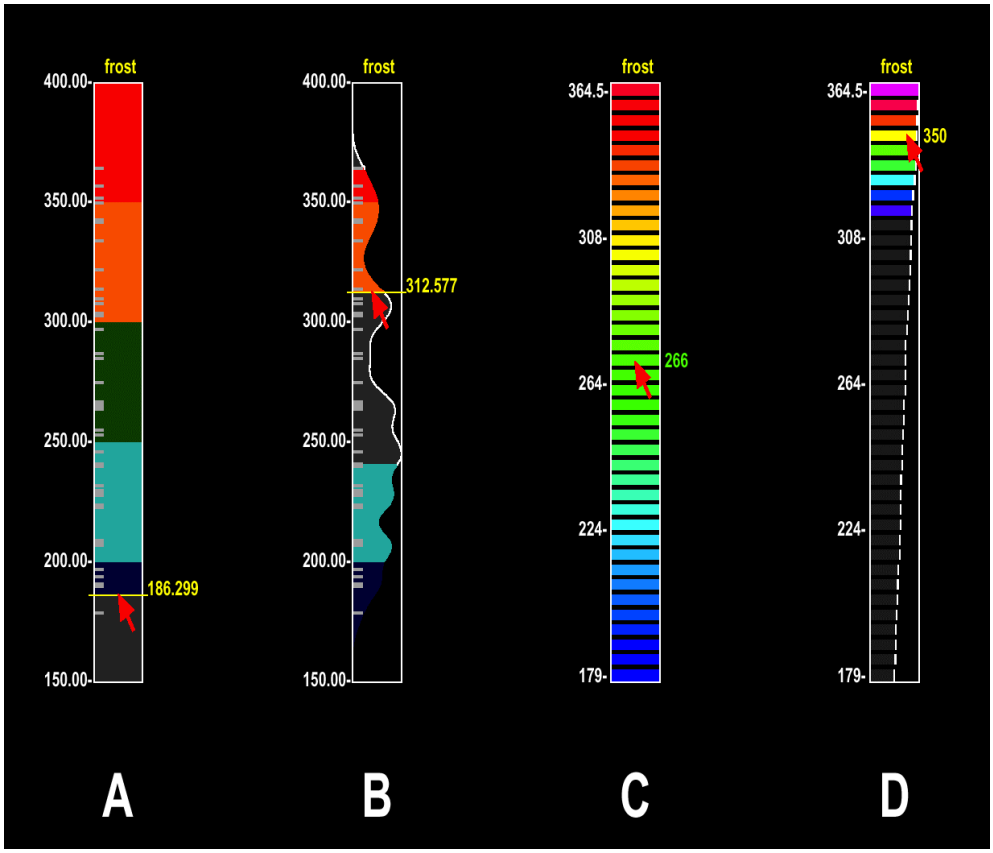

Abbildung 16 – Data Visualization Slider mit 4 Betriebsmodi

Der Data Visualization Slider erlaubt sowohl die Auswahl von Bereichen, als auch die Mehrfachselektion. Zusätzlich dient die Oberfläche des Data Visualization Slider dazu, die Verteilung der zugrunde

liegenden Daten zu visualisieren. Außerdem informiert der Data Visualization Slider den Anwender mittels einer Skala über den verwendeten Wertebereich.

Der Data Visualization Slider unterscheidet sich optisch sehr stark vom klassischen Schieberegler. Während der klassische Schieberegler aus einem „Knauf“ (an dem „gezogen“ wird) und einem „Hintergrund“ (über den sich der „Knauf“ bewegt) besteht, erscheint der Data Visualization Slider als eine einzige, rechteckige, teilweise bunt gefärbte Fläche.

Auch die Vorgehensweise des Anwenders beim Arbeiten mit dem Data Visualization Slider unterscheidet sich etwas von der des klassischen Schiebereglers. Die Vorgehensweise bei Benutzung des Modus „C“ kann – zerlegt in vier Arbeitsschritte - z.B. wie folgt aussehen:

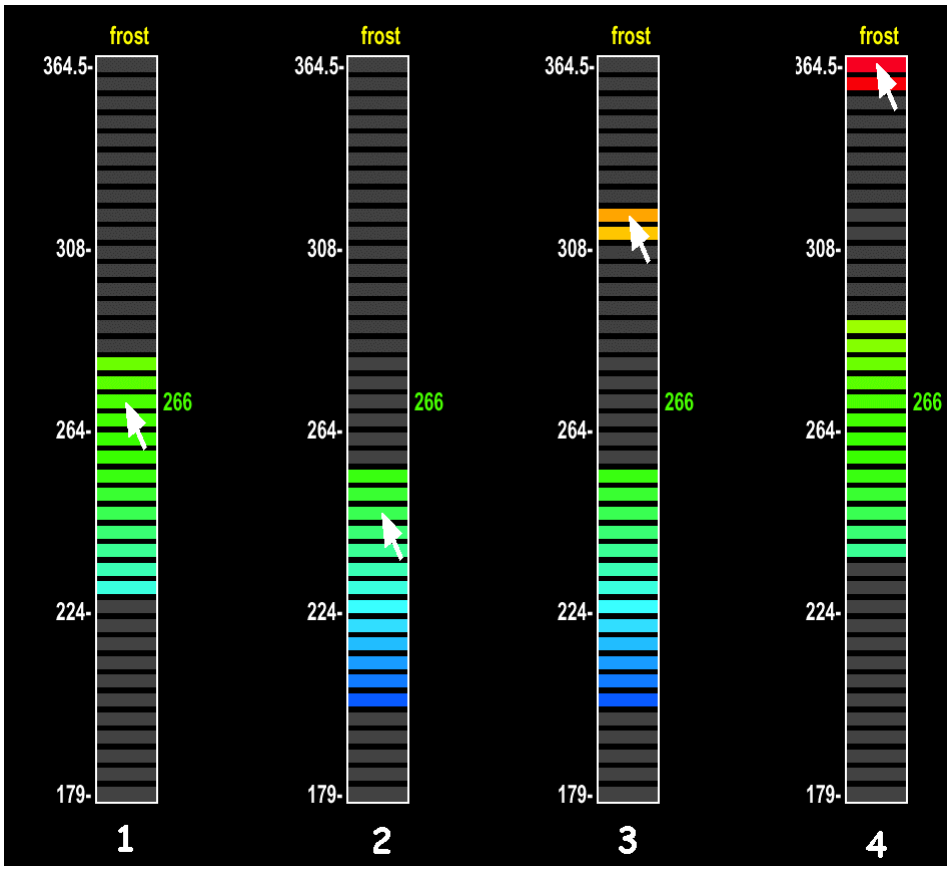

Abbildung 17 - Funktionsweise Data Visualization Slider

1. Zustand 1 ist der Startzustand des Data Visualization Slider. Es sind einerseits dunkelgrau, und andererseits bunt gefärbte Segmente zu erkennen. Die bunt gefärbten Segmente stellen den „Knauf“ des klassischen Schiebereglers dar, während die dunkelgrau gefärbten Segmente den „Hintergrund“ des klassischen Schiebereglers darstellen.

2. Der Anwender verschiebt nun den „Knauf“, indem er die Maus darauf bewegt, die Maustaste drückt (ohne sie loszulassen), dann den „Knauf“ an die gewünschte Position bewegt, und die Maustaste wieder loslässt. Das Vorgehen entspricht bisher also dem des klassischen Schiebereglers. Damit ist nun Zustand 2 erreicht.

3. Der Anwender macht nun Gebrauch von der Möglichkeit der Mehrfachselektion – er wählt koexistent zum bereits existierenden Bereich einen zweiten Bereich aus, indem er zwei bislang dunkelgrau gefärbte Segmente mit der Maus anklickt. Der „Knauf“ hat dadurch seine Erscheinung verändert, und damit ist Zustand 3 erreicht.

4. Der Anwender verschiebt nun erneut den „Knauf“, mit dem gleichen Vorgehen wie bereits in Zustand 2.

Nachdem nun geklärt ist, wie das Arbeiten mit dem Data Visualization Slider grundsätzlich abläuft, steht noch die Frage im Raum, worin sich die vier Betriebsarten voneinander unterscheiden. Die vier Betriebsarten des Data Visualization Slider unterscheiden sich voneinander darin, wie die Datenverteilung visualisiert wird.

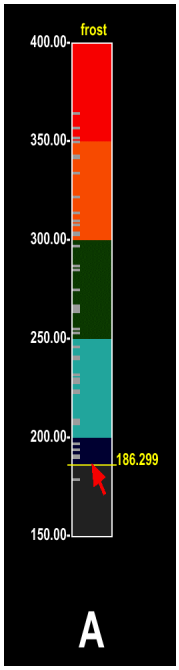

Abbildung 18 – Data Visualization Slider

1. Betriebsart A:

In der Betriebsart A wird die Datenverteilung nicht visualisiert. Zwar wird der „Knauf“ eingefärbt, aber die verwendete Farbe steht in keiner Beziehung zur Datenverteilung. Sie dient lediglich der besseren Orientierung während des Arbeitens mit dem Steuerelement.

In der Betriebsart A werden keine diskreten Werte angezeigt. Statt dessen wird dem Anwender suggeriert, dass er es mit kontinuierlichen Werten zu tun hat – er kann in beliebig kleinen Einheiten[7] markieren und verschieben.

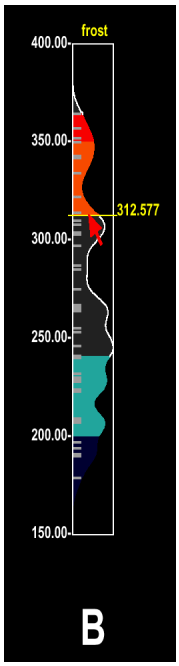

Abbildung 19 - Data Visualization Slider

2. Betriebsart B:

Betriebsart B entspricht weitestgehend der Betriebsart A, allerdings mit dem feinen Unterschied, dass die Datenverteilung visualisiert wird:

Innerhalb des Data Visualization Slider, d.h. in dem Bereich, der eingefärbt wird, wird die Datenverteilung, d.h. die Anzahl der Datensätze, die dieser Schiebereglerposition entsprechen, als Kurve dargestellt.

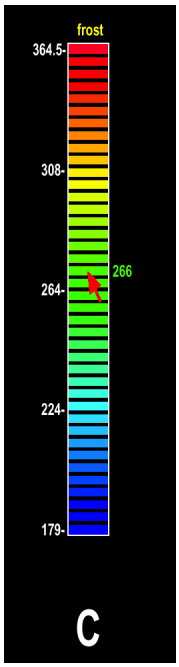

Abbildung 20 – Data Visualization Slider

3. Betriebsart C:

Betriebsart C entspricht Betriebsart A mit dem Unterschied, dass der Wertebereich diskret dargestellt wird:

Der Anwender kann lediglich diskrete Werte auswählen, und verschieben.

Für einen „sehr großen“ Wertebereich geht Betriebsart C optisch in Betriebsart A über.

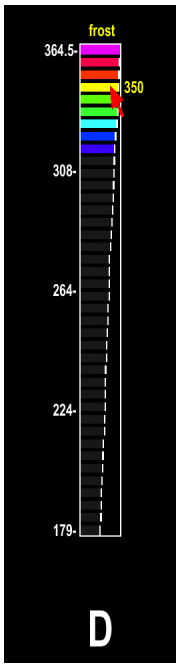

Abbildung 21 - Data Visualization Slider

4. Betriebsart D:

Die Unterschiede zwischen Betriebsart C und Betriebsart D entsprechen den Unterschieden zwischen Betriebsart A und Betriebsart B.

Innerhalb des Data Visualization Slider, d.h. in dem Bereich, der eingefärbt wird, wird die Datenverteilung, d.h. die Anzahl der Datensätze, die dieser Schiebereglerposition entsprechen, als Kurve dargestellt.

[7] „Beliebig klein“ ist technisch natürlich nicht möglich. Gemeint ist hier, dass zumindest zumindest pixelweise selektiert und verschoben werden kann.

Daher:

	Platz	Diskrete Werte	Bereich-Auswahl	Mehrfach-Selektion	„browsen“	Visualisierung	Komfort
Data Visualization Slider	+		+	+	+	+	+

6.3 Das Steuerelement “2D Widget“

Das 2D Widget ist ein von Ben Shneiderman entwickeltes Steuerelement mit der Funktionalität eines zweidimensionalen Schiebereglers:

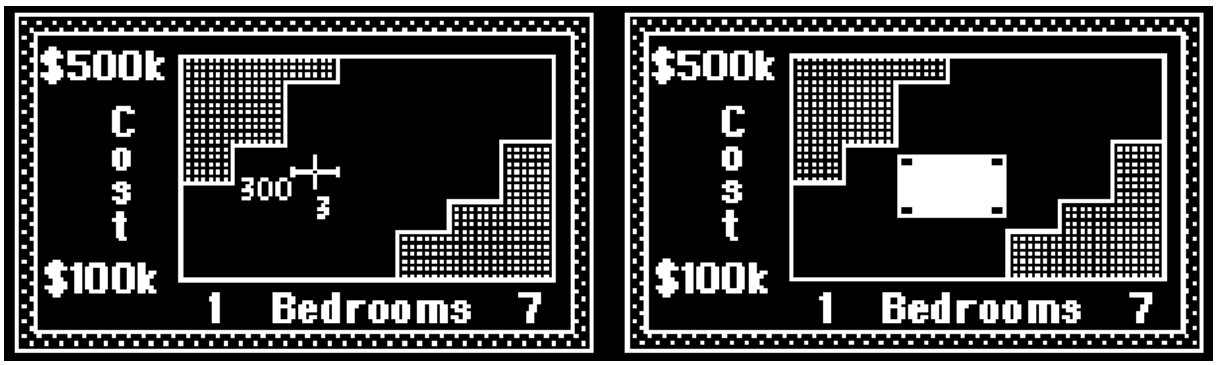

Abbildung 22 - 2D Widget

Der Anwender benutzt das 2D Widget wie die Funktion zum Zeichnen eines Rechteckes in einem Zeichenprogramm: Er zieht bei gedrückter Maustaste ein Rechteck auf, und lässt danach erst die Maustaste wieder los. Das so „gezeichnete“ Rechteck entspricht dem ausgewählten Wertebereich.

In Abbildung 22 dient ein 2D Widget dazu, eine Eingabe über die Anzahl der Schlafzimmer und den Kaufpreis einer Immobilie zu machen. Der Anwender hat in diesem Beispiel definiert, dass er für eine Immobilie mit zwei Schlafzimmern bis zu 200.000$, und für eine Immobilie mit vier Schlafzimmern bis zu 300.000$ zu zahlen bereit ist.

Das 2D Widget ist in der Lage, seinen eigenen Wertebereich zu visualisieren: Dazu werden an den beiden Achsen jeweils zwei Beschriftungen angezeigt. Außerdem existiert innerhalb des Zeichenbereiches eine schraffierte Fläche, die nicht ausgewählt werden kann: In dem Beispiel in Abbildung 22 beispielsweise beträgt der niedrigste Kaufpreis für eine Immobilie mit sieben Schlafzimmern 310.000$ - es ist daher beispielsweise nicht möglich, die Kombination „sieben Schlafzimmer“ und „100.000$“ zu wählen, da dieser Bereich schraffiert dargestellt ist.

Das 2D Widget bietet neben den bereits genannten Aspekten den Vorteil, dass der Anwender in annähernd der gleichen Zeit, in der er mittels eines Schiebereglers eine einzige Größe bestimmt, nun gleich zwei Größen bestimmen kann. Diese Möglichkeit erscheint umso sinnvoller, wenn man sie im Zusammenhang mit zweidimensionalen Visualisierungsmethoden, wie z.B. dem *Starfield-Display* betrachtet.

Das 2D Widget besitzt einen konstanten Platzbedarf, unterstützt Bereichauswahl und Mehrfachselektion. Weiterhin ermöglicht das 2D Widget das „browsen“, erlaubt aber leider nicht die Auswahl von diskreten Werten.

Daher:

	Platz	Diskrete Werte	Bereich-Auswahl	Mehrfach-Selektion	„browsen“	Visualisierung	Komfort
2D Widget	+		+	+	+		+

6.4 Zusammenfassung

Zusammenfassend ergibt sich für die klassischen und die neuartigen Steuerelemente das folgende Bild:

	Platz	Diskrete Werte	Bereich-Auswahl	Mehrfach-Selektion	„browsen“	Visualisierung	Komfort
Slider	+				+		+
Checkbox		+	+	+			+
Radiobutton		+					+
Combobox	+	+					
Alphaslider	+		+		+	+	+
Data Visualization Slider	+		+	+	+	+	+
2D Widget	+		+	+	+		+

Summa summarum: Die neuartigen Steuerelemente versprechen neue Möglichkeiten, aber noch immer gilt: Es gibt viel zu tun. Solange das folgende Steuerelement nicht entwickelt worden ist, darf das Ziel dieser Forschung nicht als „erreicht“ bezeichnet werden:

	Platz	Diskrete Werte	Bereich-Auswahl	Mehrfach-Selektion	„browsen“	Visualisierung	Komfort
???????????	+	+	+	+	+	+	+

7 Die Lösung (2): Magic Lenses

7.1 Einleitung

In den vorangegangenen Kapiteln wurden die Herausforderungen erklärt, die *Dynamic Queries* an Steuerelemente stellen, und die Probleme aufgezeigt, beim Versuch, ihnen durch die Erweiterung bestehender Steuerelemente zu begegnen.

In diesem Kapitel wird nun ein alternativer Lösungsansatz beschrieben, der von Ken Fishkin entwickelt worden ist, und auf den klangvollen Namen „*Magic Lenses*“ hört, manchmal auch „*Movable Filters*“ genannt.

Hierbei schiebt der Anwender mehrere sogenannte „Linsen“ übereinander, die er selber definieren kann, und die jeweils einem unkomplizierten, booleschen Ausdruck entsprechen. Auf diese Art und Weise kann der Anwender spielend beliebig komplizierte Datenbankabfragen gestalten.

7.2 Linsen als Filter

Die Idee das *Magic Lenses* ist unheimlich einfach, und doch sehr wirkungsvoll. Die Funktionsweise lässt sich am Besten an einem Beispiel erklären:

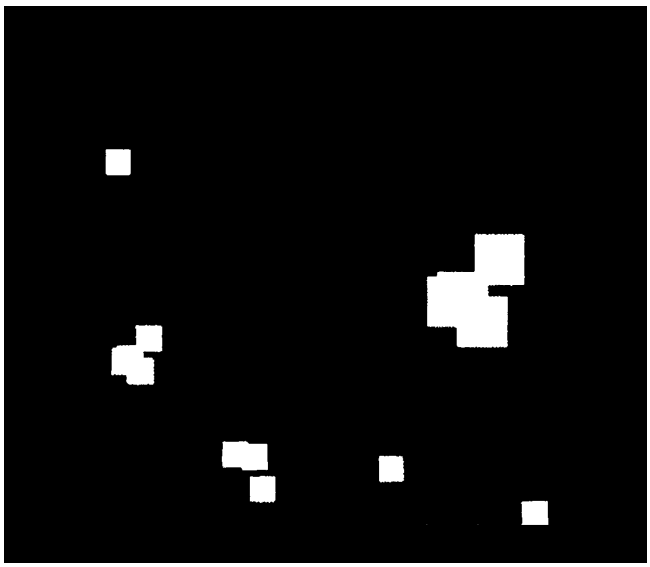

Abbildung 23 - Magic Lenses - SELECT * FROM HOMES

Abbildung 23 zeigt eine Menge weißer Quadrate auf schwarzem Hintergrund, diese Anordnung lässt sich am Besten als *Starfield-Display* umschreiben. Jedes weiße Quadrat symbolisiert einen gefundenen Datensatz, die in diesem Beispiel wieder einzelnen Immobilien entsprechen.

Der Anwender verfeinert nun seine Abfrage, indem er Stück für Stück weitere Abfragekriterien definiert. Er erzeugt eine Linse, bindet sie an die Größe „Kaufpreis“, und setzt ihre Filterfunktion auf „> 225.820“. Diese Linse schiebt er über den in Abbildung 23 dargestellten Bereich:

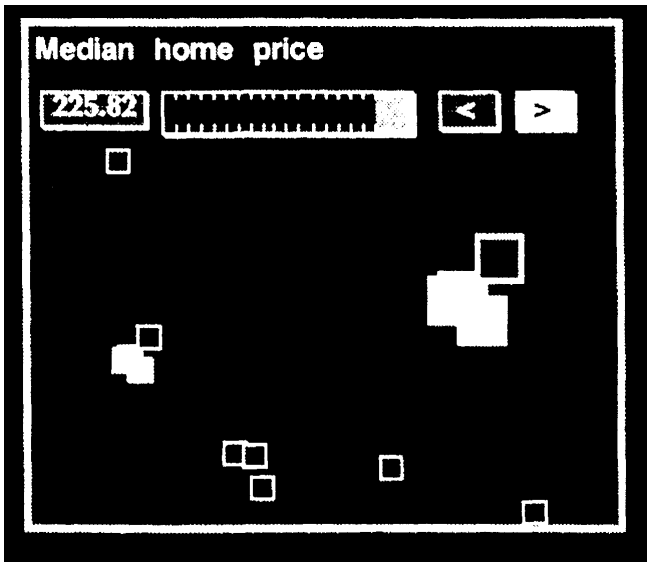

Abbildung 24 - Magic Lenses - Filterfunktion

Die Konsequenz ist, dass alle Immobilien, die diesen Filter nicht „passieren“, durch die Linse nicht mehr als ausgefüllte, sondern als nicht-ausgefüllte, weiße Quadrate dargestellt werden.

7.3 Details On Demand

Findet der Anwender nun Gefallen an einer der Immobilien, und möchte gerne nähere Informationen über diese erlangen, so klickt er einfach mit der Maus auf das entsprechende Symbol:

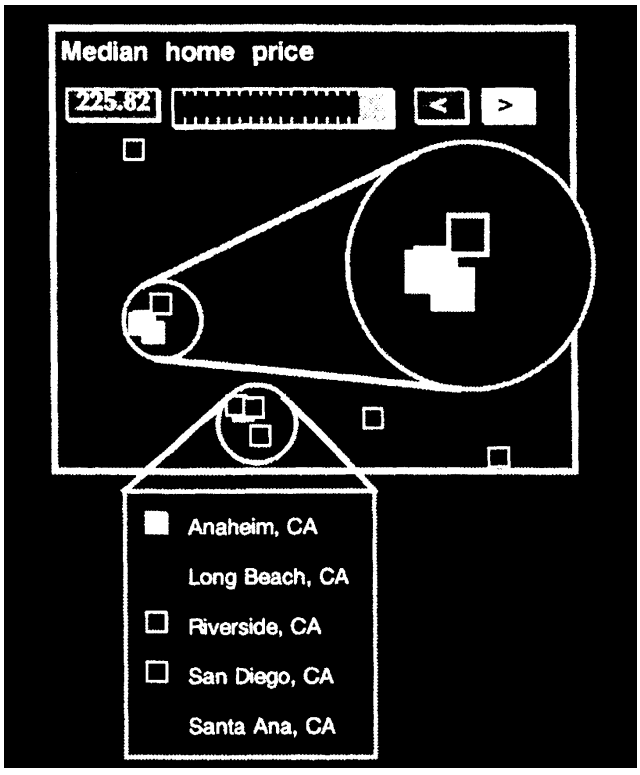

Abbildung 25 - Magic Lenses - Details On Demand

Die detaillierten Informationen über die selektierten Symbole werden dann eingeblendet. Diese Technik nennt man auch *Details-On-Demand*.

7.4 Verwendung von mehr als einer Linse

Der Anwender kann beliebig viele solcher Linsen definieren, und frei auf dem Bildschirm verschieben; die Darstellung wird dabei in Echtzeit an die neue Linsenposition angepasst. Schiebt der Anwender zwei (oder mehr) Linsen (partiell) übereinander, so wird die Wirkung der entsprechenden Linsen miteinander verknüpft – ganz so, wie dies auch bei optischen Linsen der Fall wäre:

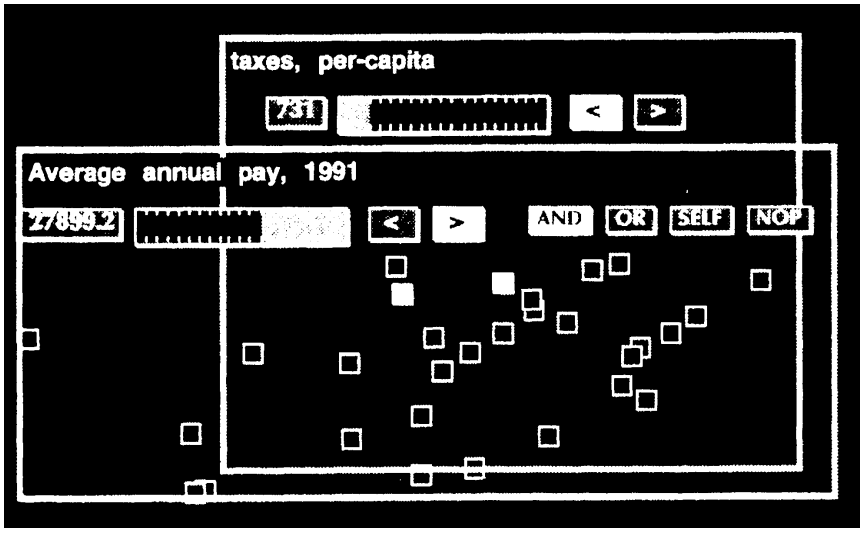

Abbildung 26 - Magic Lenses - "und" Verknüpfung

In Abbildung 27 hat der Anwender zwei Linsen übereinander geschoben. Die Folge ist, dass nur noch die Datensätze sichtbar sind, die den Bedingungen beider Linsen genügen. Alle anderen Datensätze werden wieder als nicht-ausgefüllte, weiße Quadrate dargestellt.

Der Anwender kann für jede Linse wählen, ob ihr Abfrageergebnis mit dem der darunterliegenden Linsen per „und-Verknüpfung“ (AND) oder per „oder-Verknüpfung“ (OR) verknüpft werden soll, ob die Linse wirkungslos sein soll (NOP), oder ob alle dahinterliegenden Linsen wirkungslos sein sollen (SELF).

Betätigt der Anwender nun die Schaltfläche „OR“, so ändert sich die Darstellung ohne Zeitverzug wie folgt:

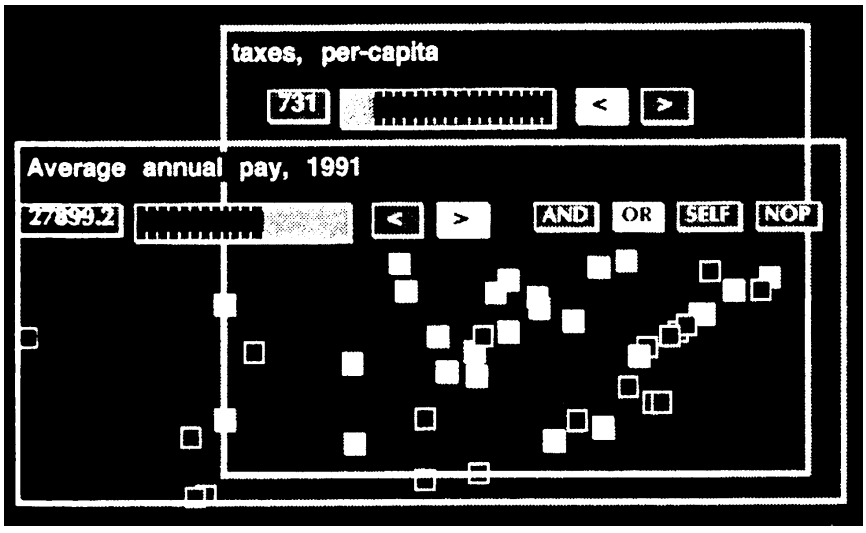

Abbildung 27 - Magic Lenses - "oder" Verknüpfung

Man sieht, wie die Anzahl der ausgefüllten Quadrate, d.h. die Anzahl der gefundenen Datensätze, merklich angestiegen ist.

7.5 Gruppierung

Es besteht die Möglichkeit, die Kombination von zwei oder mehr Linsen zu einer „Gesamtlinse“ zusammenzufassen, d.h. zu gruppieren. Diese „Gesamtlinse“ kann dann wie eine einzelne Linse behandelt werden, d.h. frei auf dem Bildschirm verschoben werden.

Bezogen auf die Boolesche Algebra stellt diese Möglichkeit des Gruppierens von Linsen die Klammerung von Booleschen (Teil-) Ausdrücken dar.

7.6 Fuzzy Logic

Es ist der Charakter und zugleich das typische Problem der klassischen Datenbankabfrage, dass der Anwender die und nur die Datensätze sieht, nach denen er mittels eines Booleschen Ausdruckes gefragt hat. Datensätze, die „knapp daneben“ liegen, werden genauso wenig zurückgegeben, wie Datensätze, die „total daneben“ liegen.

Die *Magic Lenses* bringen hier die interessante Möglichkeit der „*Fuzzy Logic*“ ins Spiel: Standardmäßig arbeiten alle Linsen mit UND- und ODER-Operationen, und den binären Operanden „0“ und „1“. Der Anwender kann den Modus der Linsen auf „*Fuzzy Logic*“ umschalten, was bewirkt, dass fortan mit Multiplikation und Addition als Operationen, und reellen Zahlen als Operanden gearbeitet wird.

Eine als Filter eingesetzte Linse besitzt nun nicht länger die Rückgabewerte „0“ und „1“, sondern jeden reellen Wert zwischen „0“ und „1“ – je nachdem, wie stark der entsprechende Datensatz die definierte Bedingung verfehlt hat.

In der folgenden Abbildung sind vier Datensätze in der Detailansicht zu sehen, von denen ohne die *Fuzzy Logic* nur ein einzelner Datensatz („Miami“) die Abfragefilter bis zum Ende durchlaufen hätte. Durch die Verwendung der *Fuzzy Logic* kann der Anwender sehen, dass es einen zweiten Datensatz „Tampa“ gibt, der nur um Haaresbreite neben dem Auswahlkriterium liegt:

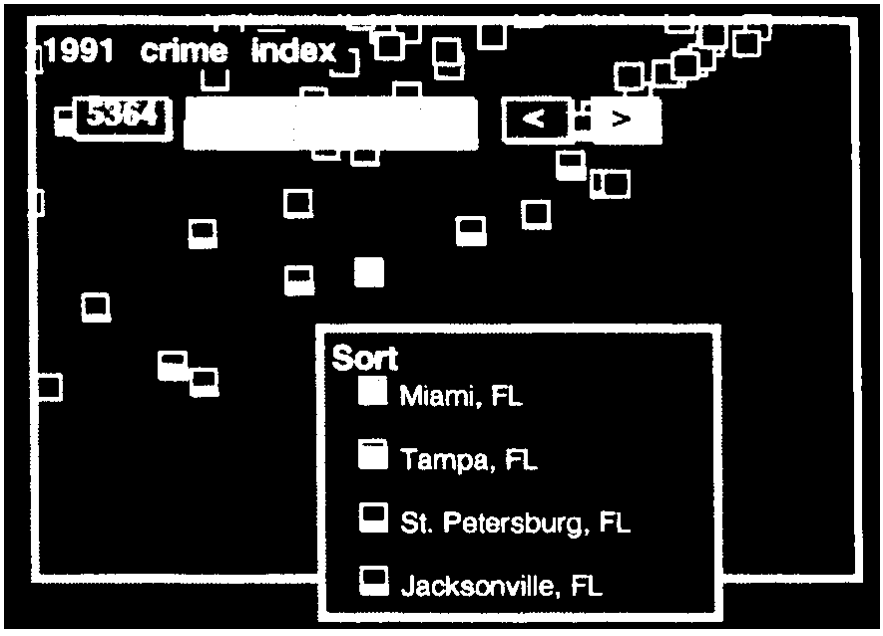

Abbildung 28 - Magic Lenses - Fuzzy Logic

7.7 Missing Values

Ein weiteres, typisches Problem der klassischen Datenbankabfrage ist das Fehlen von Werten, das sogenannte „*Missing Values Problem*“ oder „*Missing Data Problem*“. Wenn der Datenbestand z.B.

auf empirischen Messwerten beruht, kann es gut sein, dass einige Werte fehlen; weil sie sich nicht messen lassen, oder das Messen zu teuer ist.

Bei Verwendung einer klassischen Datenbankabfrage fallen Datensätze mit solchen, fehlenden Werten zumeist unter den Tisch. Durch die Verwendung einer „Missing Data“-Linse räumen die *Magic Lenses* mit diesem Problem auf:

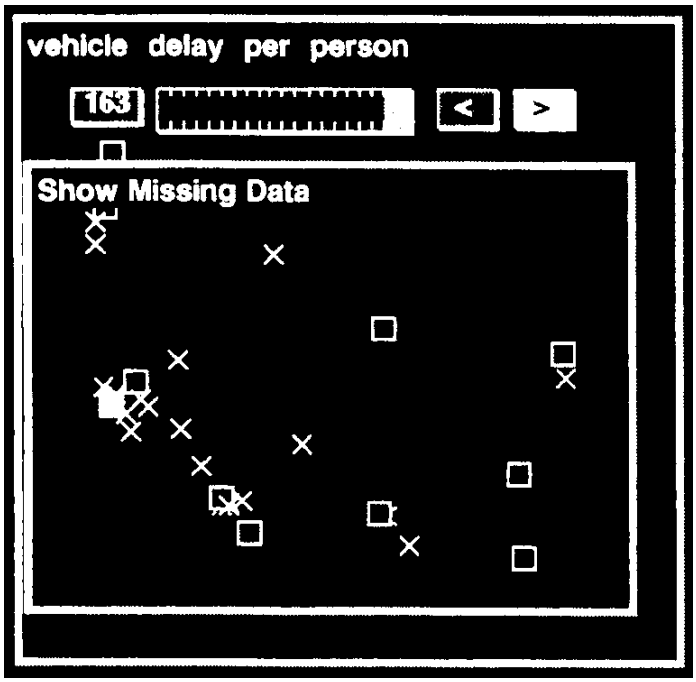

Abbildung 29 - Magic Lenses - Missing Values

Der Anwender “schiebt” die „Missing Data“-Linse wie jede andere Linse über die entsprechende Bildschirmposition, und alle Datensätze, die aufgrund eines fehlenden Wertes an einer der dahinterliegenden Linsen gescheitert sind, werden als „X“ dargestellt.

8 Zusammenfassung und Ausblick

Dynamic Queries sind mächtig, aber nicht allmächtig. Die neuen, verspielten Möglichkeiten gegenüber den traditionellen Abfragesprachen sind verlockend, aber sie stellen auch neue Herausforderungen, denen es zu begegnen heißt. Es gilt neue Steuerelemente, Benutzeroberflächen, Datenformate, Algorithmen, Protokolle und Schnittstellen zu entwerfen.

Es ist gezeigt worden, dass auch die neuartigen Steuerelemente *Alphaslider*, *Data Visualization Slider* und *2D Widget* nicht in der Lage sind, allen Anforderungen der im Sinne von *Dynamic Queries* / *Tight Coupling* entwickelten Anwendungen zu genügen; und dass es noch zahlreiche Hürden zu überwinden gilt, bis die Visionen der zitierten Vorreiter in alltägliche Anwendungssoftware Einzug halten.

Alle in diesem Papier vorgestellten Anwendungen der *Dynamic Queries* sind individuell angefertigte, und auf ein bestimmtes Problem zugeschnittene Speziallösungen. Zukünftig müssen allgemein einsetzbare Werkzeuge um die Fähigkeiten der *Dynamic Queries* erweitert werden, wie z.B. bestehende Datenbanksysteme und Tabellenkalkulationen – sogenannte „Standardsoftware“. Insbesondere fehlen auch Entwicklungswerkzeuge.

Körperlich behinderte Personen mit Sehschwächen und/oder motorischen Problemen werden durch *Dynamic Queries* in der Form, wie sie in diesem Papier präsentiert worden sind, mehr Nachteile als Vorteile erwachsen. Hier gilt es, eine für alle Anwender gleichermaßen nützliche Kompromisslösung zu entwickeln. Denkbar wäre z.B. die Einbeziehung von Audiosignalen in die „Visualisierung“.

Noch immer gilt, dass relativ komplexe Datenbankabfragen wie z.B. „group by“ nur durch die klassische Datenbankabfrage formuliert werden können. Hier gilt es, noch universellere und vielfältigere Benutzerinterfaces und Steuerelemente zu entwickeln, um mehr und mehr auf die Kommandozeile verzichten zu können. Als vorläufiges Ziel könnte hier die Erreichung einer Universalität wie z.B. der von *SQL* stehen.

9 Kommentiertes Literaturverzeichnis

[1] Dynamic Queries

(Dynamische Datenbankabfragen)

Card, Stuart K. and Mackinlay, Jock D. and Shneiderman, Ben, Readings In Information Visualization - Using Visions To Think, Morgan Kaufmann, 1999, ISBN 1-55860-533-9

Das Sterben nicht-visueller Dienstprogramme ist allgegenwärtig, und macht auch vor der Datenbankabfrage nicht halt. Die technologischen Fortschritte in Bezug auf Massenspeicher und Netzwerke haben zu einem Overkill an Daten geführt, der für jeden Einzelnen schwer zu bewältigen ist. Neue Abfragetechniken versprechen eine Besserung: Kommandozeilen-Werkzeuge und kryptische SQL-Ausdrücke weichen den Echtzeit-Simulationen in der virtuellen Realität. Eine Datenbankabfrage wird nicht mehr gestartet, und läuft bis zu ihrem „Ende“, sondern verlässt den Begriff des „Algorithmus“: eine kontinuierliche Interaktion zwischen dem Anwender und der ausgefeilten Oberfläche vermittelt dem Anwender das Gefühl, an der Abfrage teilzunehmen, oder ein Teil der Abfrage zu sein. Diese neuen Methoden profitieren elegant von den kognitiven Fähigkeiten des menschlichen Auges.

[2] Dynamic Queries for Visual Information Seeking

(Dynamische Datenbankabfragen zur Informationssuche)

Shneiderman, Ben, Dynamic Queries for visual information seeking, IEEE Software 11, 6 (1994), 70-77.

Dieser Artikel berichtet über Pro und Contras von dynamischen Datenbankabfragen gegenüber traditionellen Abfragesprachen wie *SQL*. Dynamische Datenbankabfragen versprechen kürzere Einarbeitungszeiten, intuitivere und weniger fehleranfällige Möglichkeiten der Abfrageformulierung und eine ansprechendere Präsentation des Abfrageergebnisses; sie diesen dem Neuling ebenso wie dem Routinier. Dynamische Datenbankabfragen bedeuten aber auch höhere Anforderungen an grafische Ressourcen, neue Herausforderung für Datenbankstrukturen und –Schnittstellen, und Bedarf für die Entwicklung neuer Algorithmen.

[3] Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays

(Visuelle Informationssuche: Die Beziehung zwischen dynamischen Datenfiltern und Starfield-Anzeigen)

4) Ahlberg, Christopher and Shneiderman, Ben, Visual Information Seeking: Tight coupling of dynamic query filters with starfield displays, Proc. of ACM CHI94 Conference (April 1994), 313-317 + color plates. Reprinted in Baecker, R. M., Grudin, J., Buxton, W. A. S., and Greenberg, S. (Editors), Readings in Human-Computer Interaction: Toward the Year 2000, Second Edition, Morgan Kaufmann Publishers, Inc., San Francisco, CA (1995), 450-456.

Dieser Artikel erklärt die als „*tight coppling*“ (enge Verbindung) bezeichnete Beziehung zwischen der Datenbankabfrage und dem Abfrageergebnis. Die traditionelle Unterscheidung im Sinne von Ursache und Reaktion wird durch neuartige Abfrageoberflächen mit Echtzeitverhalten weitestgehend aufgehoben. Steuerelemente greifen der Datenabfrage voraus, indem sie ihren Wertebereich dynamisch anpassen; umgekehrt dient jedes Abfrageergebnis als möglicher Ausgangspunkt für weitere Abfragen. Durch simple Mausklicks werden Abfragen kontinuierlich präzisiert und neu formuliert, der Abfragevorgang wandelt sich dadurch von einem Frage-Antwort-Dialog in ein „browsen“. Hierbei spielen geeignete Datenanzeigen und Steuerelemente eine besondere Rolle; die Autoren heben dabei vor allem die „Starfield-Anzeige“ und den „Alphaslider“ hervor, und demonstrieren deren Funktionsweise anhand der selbstentwickelten Applikationen „FilmFinder“ und „*Dynamic HomeFinder*“.

[4] Data Visualization Sliders

Scrollbalken zur Datenvisualisierung

S. G. Eick, AT&T Bell Laboratories, Quelle #1, Seiten 251-252)

Der klassische Schieberegler ist seit Urbeginn ein fester Bestandteil der grafischen Benutzeroberflächen. S. G. Eick stellt in diesem Artikel eine Weiterentwicklung dieses Steuerelementes vor, speziell getrimmt auf den Einsatz in Dynamischen Datenbankabfragen. Ganz im Sinne von „Tight Coupling“ wird das Ergebnis der Datenbankabfrage direkt innerhalb des Schieberegler präsentiert. Trotz gleichem Platzbedarf bietet der erweiterte Schieberegler Möglichkeiten wie Bereichsauswahl und Mehrfachselektion.

[5] Enhanced Dynamic Queries via Movable Filters

Erweiterte Datenabfrage mit beweglichen Filtern

K. Fishkin und M. C. Stone, Xerox PARC, Quelle #1, Seiten 253-259)

Dieser Artikel beschreibt den Spagat zwischen dem obersten Gebot „intuitive Bedienung“ und dem Wunsch, damit alle denkbaren Datenbankabfragen formulieren zu können. Die Autoren stellen sich dieser Problematik, und führen in die Idee der „beweglichen Filter“ oder auch „magischen Linsen“ ein. Bei diesem Prinzip der Datenbankabfrage definiert der Anfragesteller auf der Oberfläche „Linsen“, die einem booleschen Ausdruck entsprechen. Diese Linsen können beliebig übereinandergeschoben, neudefiniert, gruppiert und wieder gelöscht werden. Trotz dieser einfachen und verspielten Benutzerführung ist auf diese Art und Weise jede Datenbankabfrage realisierbar.

[6] Dynamic HomeFinder Project Plan Section 1: Introduction and Overview

„Dynamic HomeFinder“ Projektplanung, Teil 1: Einführung und Übersicht

Christopher Williamson and Tom Smallwood, 1995

Christopher Williamson and Tom Smallwood, die Autoren der erweiterten Version von “*Dynamic HomeFinder*” erklären hier die Ziele und Grundlagen ihres Softwareprojektes, und führen in die Software ein.

Dieses Dokument wurde unter der URL http://www.dqsoft.com/homefind/dhintro.htm gefunden.

10 Abbildungsverzeichnis

11 Index

S

T

U

W