4. ArcView Anwendertreffen

Datenbankanbindung mit ArcView

Daniel Fuchs

PAN Partnerschaft

Gliederung

Teil 1: Grundbegriffe zu Datenbanken

Teil 2: SQL – menügesteuerte Datenabfragen aus ArcView

Teil 3: SQL und Avenue – Einstieg in die Programmierung

Teil 4: DDE – ArcView und Datenbank arbeiten zusammen

Teil 1: Grundbegriffe Datenbanken

Datenbanken dienen zum Erstellen, Speichern und Verarbeiten von tabellarisch geordneten Daten. Diese Daten können in den verschiedensten Formaten vorliegen: zum Beispiel als numerische Werte (Zahlen), als Text (Zeichenfolge) oder als Datum.

Am Beispiel einer weit verbreiteten Datenbank, MS Access aus der Microsoft Office-Produktreihe, können die drei wichtigsten Funktionen einer Datenbank dargestellt werden:

- Datendefintion
- Datenspeicherung
- Datenmanipulation und -abfrage

Grundbegriffe: **Datendefinition - Tabellen**

Das grundlegende Format für die Defintion von Daten in Access ist die **Tabelle**.

BKArten : Tabelle

ID	ART	ARTNAME	DEUTSCH
6132193	9P01781600	Campanula persicifolia	Pfirsichblättrige Glockenblume
6132193	9P01782500	Campanula rotundifolia agg.	Artengruppe Rundblättrige Glockenblume
6132193	9P01875100	Carex flacca	Blau-Segge
6132193	9P01890400	Carlina vulgaris	Golddistel
6132193	9P01910100	Carpinus betulus	Hainbuche
6132193	9P02000700	Centaurea jacea	Wiesen-Flockenblume
6132193	9P02003400	Centaurea scabiosa	Skabiosen-Flockenblume
6132193	9P02330100	Cirsium acaule	Stengellose Kratzdistel
6132193	9P02330200	Cirsium arvense	Acker-Kratzdistel
6132193	9P02330700	Cirsium eriophorum	Wollköpfige Kratzdistel
6132193	9P02380500	Clematis vitalba	Gewöhnliche Waldrebe

Datensatz: 1 von 80114

Grundbegriffe: **Datendefinition - Tabellen**

Tabellenspalten haben einen Namen. Alle Daten einer Spalte weisen das selbe Format auf (hier Text oder Zeichenfolge).

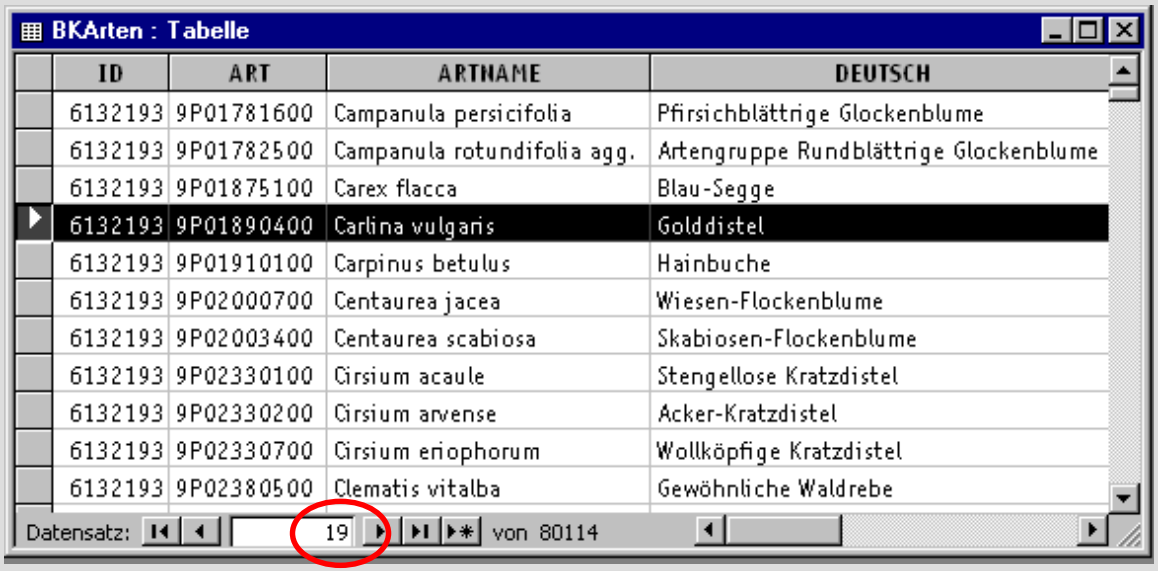
BKArten : Tabelle

ID	ART	ARTNAME	DEUTSCH
6132193	9P01781600	Campanula persicifolia	Pfirsichblättrige Glockenblume
6132193	9P01782500	Campanula rotundifolia agg.	Artengruppe Rundblättrige Glockenblume
6132193	9P01875100	Carex flacca	Blau-Segge
6132193	9P01890400	Carlina vulgaris	Golddistel
6132193	9P01910100	Carpinus betulus	Hainbuche
6132193	9P02000700	Centaurea jacea	Wiesen-Flockenblume
6132193	9P02003400	Centaurea scabiosa	Skabiosen-Flockenblume
6132193	9P02330100	Cirsium acaule	Stengellose Kratzdistel
6132193	9P02330200	Cirsium arvense	Acker-Kratzdistel
6132193	9P02330700	Cirsium eriophorum	Wollköpfige Kratzdistel
6132193	9P02380500	Clematis vitalba	Gewöhnliche Waldrebe

Datensatz: 1 von 80114

Grundbegriffe: **Datendefinition - Tabellen**

Tabellenzeilen werden auch als Datensätze bezeichnet. Sie werden in Access fortlaufend nummeriert.



BKArten : Tabelle

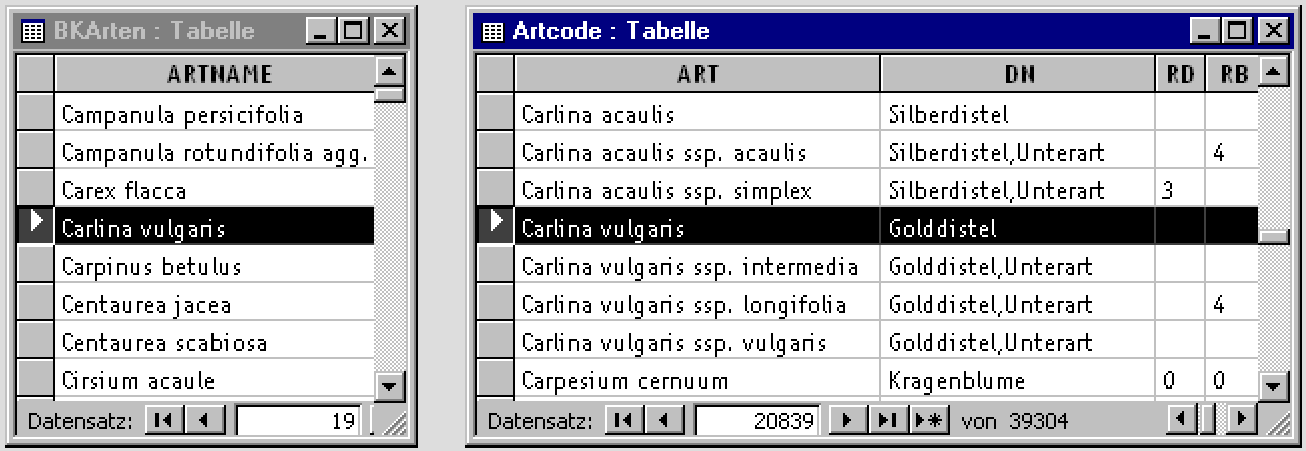
ID	ART	ARTNAME	DEUTSCH
6132193	9P01781600	Campanula persicifolia	Pfirsichblättrige Glockenblume
6132193	9P01782500	Campanula rotundifolia agg.	Artengruppe Rundblättrige Glockenblume
6132193	9P01875100	Carex flacca	Blau-Segge
6132193	9P01890400	Carlina vulgaris	Golddistel
6132193	9P01910100	Carpinus betulus	Hainbuche
6132193	9P02000700	Centaurea jacea	Wiesen-Flockenblume
6132193	9P02003400	Centaurea scabiosa	Skabiosen-Flockenblume
6132193	9P02330100	Cirsium acaule	Stengellose Kratzdistel
6132193	9P02330200	Cirsium arvense	Acker-Kratzdistel
6132193	9P02330700	Cirsium eriophorum	Wollköpfige Kratzdistel
6132193	9P02380500	Clematis vitalba	Gewöhnliche Waldrebe

Datensatz: 19 von 80114

Grundbegriffe: **Datendefinition - Tabellenbeziehungen**

Die verschiedenen in einer Datenbank enthaltenen Tabellen können in definierten Beziehungen zueinander stehen. Die beiden häufigsten Beziehungen sind dabei:

Jedem Datensatz in Tabelle 1 entspricht **genau ein** Datensatz in Tabelle 2.



BKArten : Tabelle

ARTNAME
Campanula persicifolia
Campanula rotundifolia agg.
Carex flacca
Carlina vulgaris
Carpinus betulus
Centaurea jacea
Centaurea scabiosa
Cirsium acaule

Datensatz: 19

Artcode : Tabelle

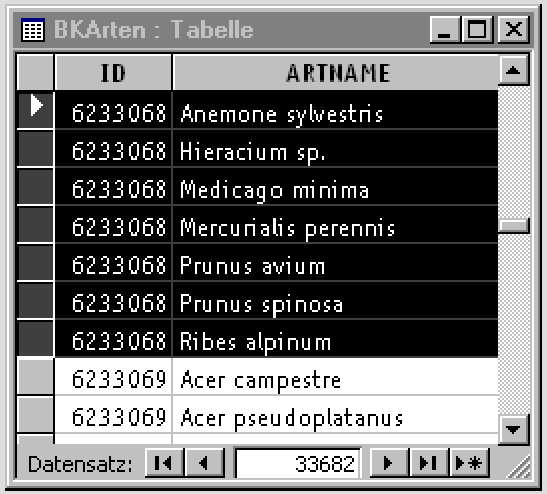
ART	DN	RD	RB
Carlina acaulis	Silberdistel		
Carlina acaulis ssp. acaulis	Silberdistel,Unterart		4
Carlina acaulis ssp. simplex	Silberdistel,Unterart	3	
Carlina vulgaris	Golddistel		
Carlina vulgaris ssp. intermedia	Golddistel,Unterart		
Carlina vulgaris ssp. longifolia	Golddistel,Unterart		4
Carlina vulgaris ssp. vulgaris	Golddistel,Unterart		
Carpesium cernuum	Kragenblume	0	0

Datensatz: 20839 von 39304

Grundbegriffe: **Datendefinition - Tabellenbeziehungen**

Die verschiedenen in einer Datenbank enthaltenen Tabellen können in definierten Beziehungen zueinander stehen. Die beiden häufigsten Beziehungen sind dabei:

Jedem Datensatz in Tabelle 1 entsprechen **mehrere** Datensätze in Tabelle 2.

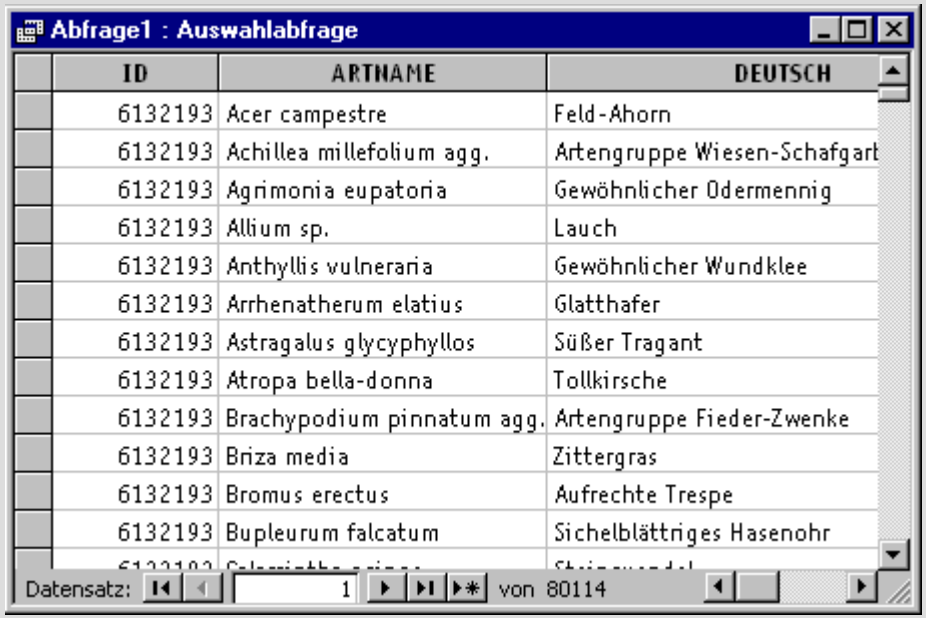


Grundbegriffe: **Datenmanipulation und -abfragen**

Für die meisten Anwendungen liegt die wichtigste Funktion einer Datenbank in der Auswertung von Daten über Abfragen.

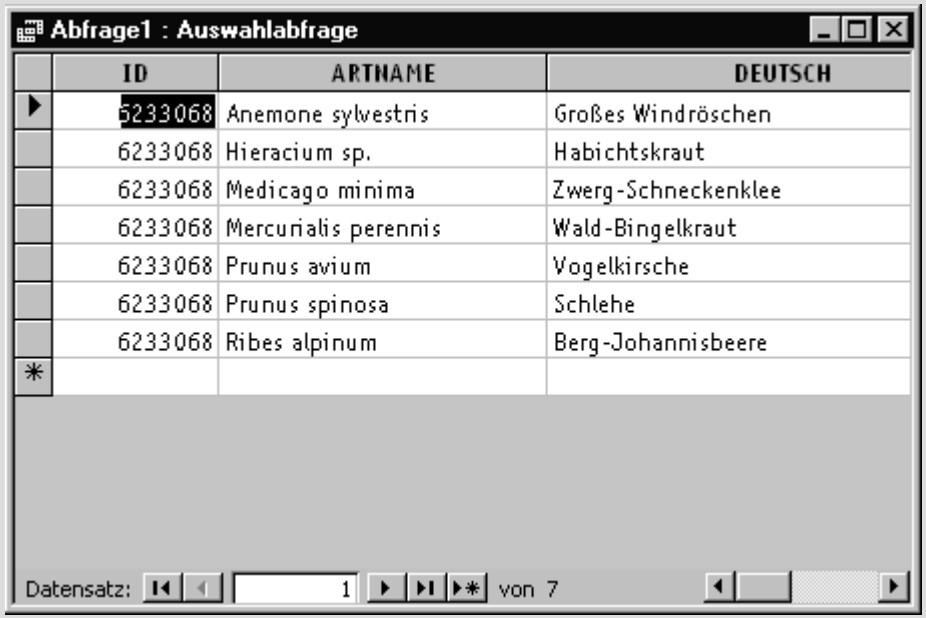
Grundbegriffe: **Datenmanipulation und -abfragen**

Durch Abfragen können Daten angezeigt, gefiltert, berechnet, summiert, gruppiert und verändert werden. Dabei können mehrere miteinander verbundene Tabellen in ein und der selben Abfrage verwendet werden.



Grundbegriffe: **Datenmanipulation und -abfragen**

Für den sinnvollen Ablauf von Abfragen ist die Angabe von Kriterien entscheidend. Durch Kriterien wird die angezeigte Datenmenge auf ein gewünschtes Teilergebnis eingegrenzt.



Grundbegriffe: **SQL**

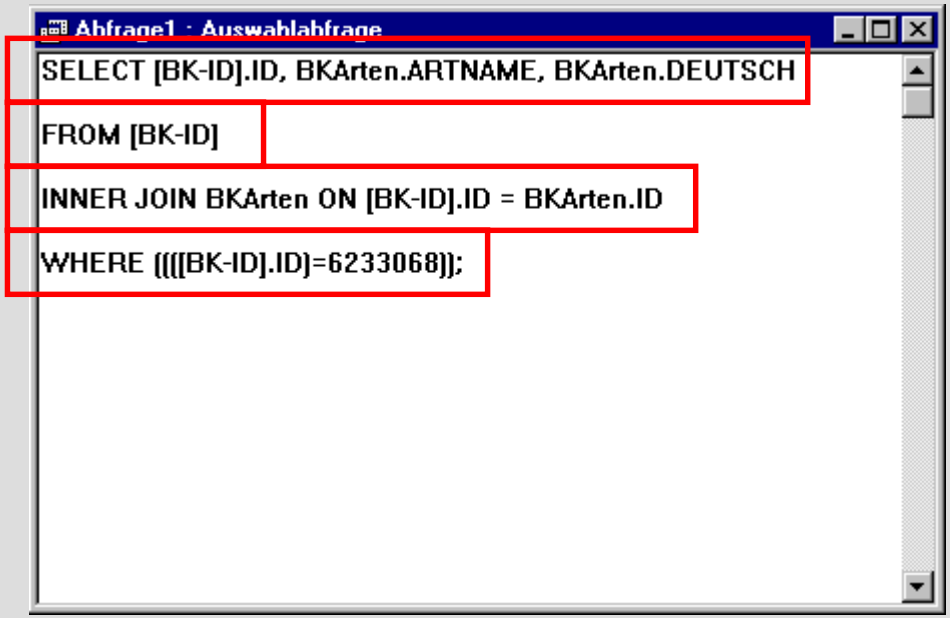
Abfragen innerhalb von Datenbanken und von anderen Anwendungen heraus auf Datenbankinhalte verwenden häufig eine spezielle Programmiersprache. Diese Sprache wird SQL genannt, was für „structured query language“ oder „strukturierte Abfragesprache“ steht.

Wer mit Access arbeitet, muss sich normalerweise nicht mit SQL beschäftigen. Abfragen werden in eigenen Abfragenfenstern erstellt und dann vom Programm selbständig in SQL übersetzt. Für den Zugriff auf eine Datenbank von anderen Programmen heraus ist aber oft die Erstellung von SQL-Befehlen notwendig.

Glücklicherweise ist SQL ziemlich leicht zu verstehen.

Grundbegriffe: **SQL**

Access bietet die Möglichkeit, den vom Programm erzeugten SQL-Befehl als Text anzuzeigen:



Grundbegriffe: **SQL - SELECT**

Eine SQL-Abfrage beginnt meist mit dem Schlüsselwort SELECT. Nach SELECT folgen alle Spalten aus den beteiligten Tabellen oder Abfragen, die angezeigt werden sollen. Die Felder werden dabei immer in der Form Tabellenname.Spaltenname angezeigt. Falls der Tabellenname Sonderzeichen enthält, wird er in eckige Klammern gesetzt.

```
SELECT [BK-ID].ID, BKArten.Name, BKArten.Deutsch
```

Grundbegriffe: **SQL - FROM**

Nach der SELECT-Zeile folgt die Anweisung FROM. In dieser Zeile werden alle Tabellen genannt, die an der Abfrage beteiligt sind.

```
SELECT [BK-ID].ID, BKArten.Name, BKArten.Deutsch
FROM [BK-ID], BKArten
```

Grundbegriffe: **SQL - FROM mit JOIN**

Nach der SELECT-Zeile folgt die Anweisung FROM. In dieser Zeile werden alle Tabellen genannt, die an der Abfrage beteiligt sind. Wenn mehrere Tabellen verknüpft sind, wird dies durch einen JOIN-Ausdruck innerhalb des FROM-Teils definiert.

```
SELECT [BK-ID].ID, BKArten.Name, BKArten.Deutsch
FROM [BK-ID] JOIN BKArten ON [BK-ID].ID = BKArten.ID
```

Grundbegriffe: **SQL - WHERE**

Mit diesen beiden Zeilen ist eine SQL-Abfrage bereits erstellt. Die Angabe von Auswahlkriterien erfolgt mit dem Schlüsselwort WHERE. Zahlenwerte werden ohne, Textwerte mit Anführungszeichen verwendet. Die Anzahl von Klammern in diesem Ausdruck wird von MSAccess recht ernst genommen.

```
SELECT [BK-ID].ID, BKArten.Name, BKArten.Deutsch
FROM [BK-ID] JOIN BKArten ON [BK-ID].ID = BKArten.ID
WHERE (((BK-ID].ID) = 63232262))
```

Grundbegriffe: **SQL - WHERE**

Abgeschlossen wird eine vollständige SQL-Abfrage immer mit einem Strichpunkt.

```
SELECT [BK-ID].ID, BKArten.Name, BKArten.Deutsch
FROM [BK-ID] JOIN BKArten ON [BK-ID].ID = BKArten.ID
WHERE (([BK-ID].ID) = 63232262));
```

Grundbegriffe: **SQL - Weitere Schlüsselwörter**

Es gibt natürlich noch zahlreiche weitere Schlüsselwörter, mit denen sich Abfrage detaillierter gestalten lassen. Eine Sortierung der ausgegebenen Daten erfolgt mit ORDER BY.

```
SELECT [BK-ID].ID, BKArten.Name, BKArten.Deutsch
FROM [BK-ID] JOIN BKArten ON [BK-ID].ID = BKArten.ID
WHERE (([BK-ID].ID) = 63232262))
ORDER BY BKArten.Name;
```

Grundbegriffe: **SQL - Weitere Schlüsselwörter**

Es gibt natürlich noch zahlreiche weitere Schlüsselwörter, mit denen sich Abfrage detaillierter gestalten lassen. Eine Sortierung der ausgegebenen Daten erfolgt mit ORDER BY. Die Daten können auch nach einem bestimmten Feld gruppiert ausgegeben werden, dies erfolgt durch GROUP BY.

```
SELECT [BK-ID].ID, BKArten.Name, BKArten.Deutsch
FROM [BK-ID] JOIN BKArten ON [BK-ID].ID = BKArten.ID
WHERE ((([BK-ID].ID) = 63232262))
ORDER BY BKArten.Name
GROUP BY [BK-ID].ID;
```

Grundbegriffe: **SQL**

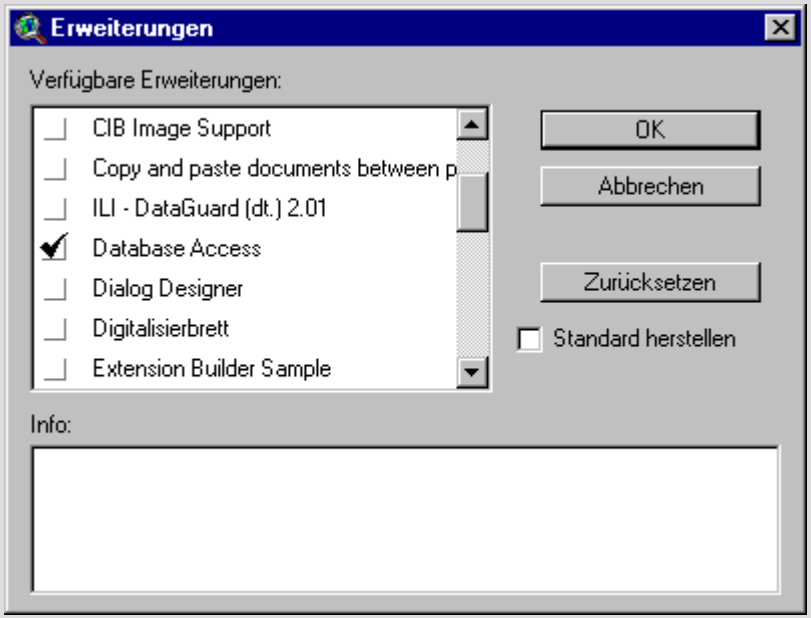
Eine genauere Definition der SQL-Schlüsselwörter findet sich in den Hilfethemen der verwendeten Programme, z. B. bei MSAccess.

Zum Üben empfiehlt es sich auf jeden Fall, menügesteuert Abfragen in MSAccess zu erstellen und sich dann den automatisch erzeugten SQL-Abfragetext genau anzusehen.

Teil 2: SQL und Arcview

Teil 2: SQL und Arcview

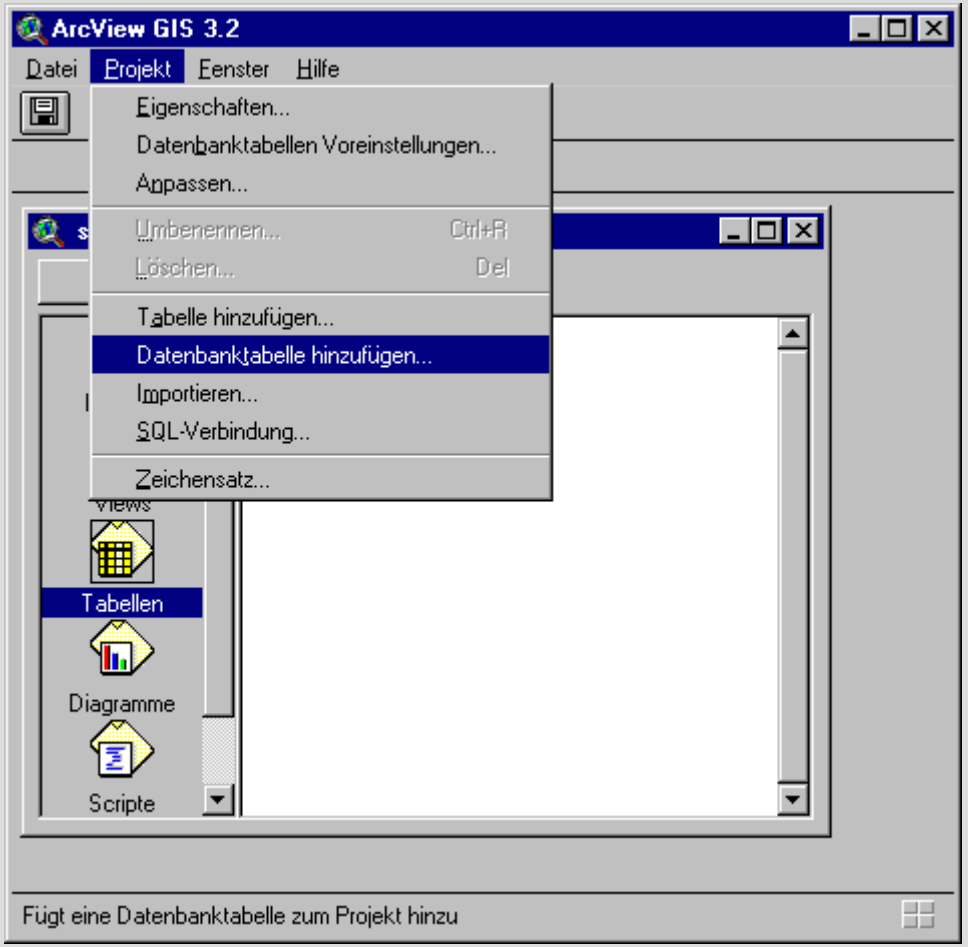
ArcView bietet die Möglichkeit, Daten aus einer Access-Datenbank direkt über SQL abzufragen und das Ergebnis tabellarisch darzustellen. Ab Version 3.2 steht dafür eine kostenlose Erweiterung zur Verfügung, der Database Access:



SQL und Arcview: **Verwendung von Database Access**

Mit Hilfe der Erweiterung „Database Access“ können alle Schritte, die zur Abfrage von Datenbank-Daten und ihrer Einbindung in ArcView notwendig sind, aus ArcView heraus durchgeführt werden.

Die Erweiterung wird mit dem Menüpunkt „Datenbanktabelle hinzufügen“ im Projektmenü gestartet.

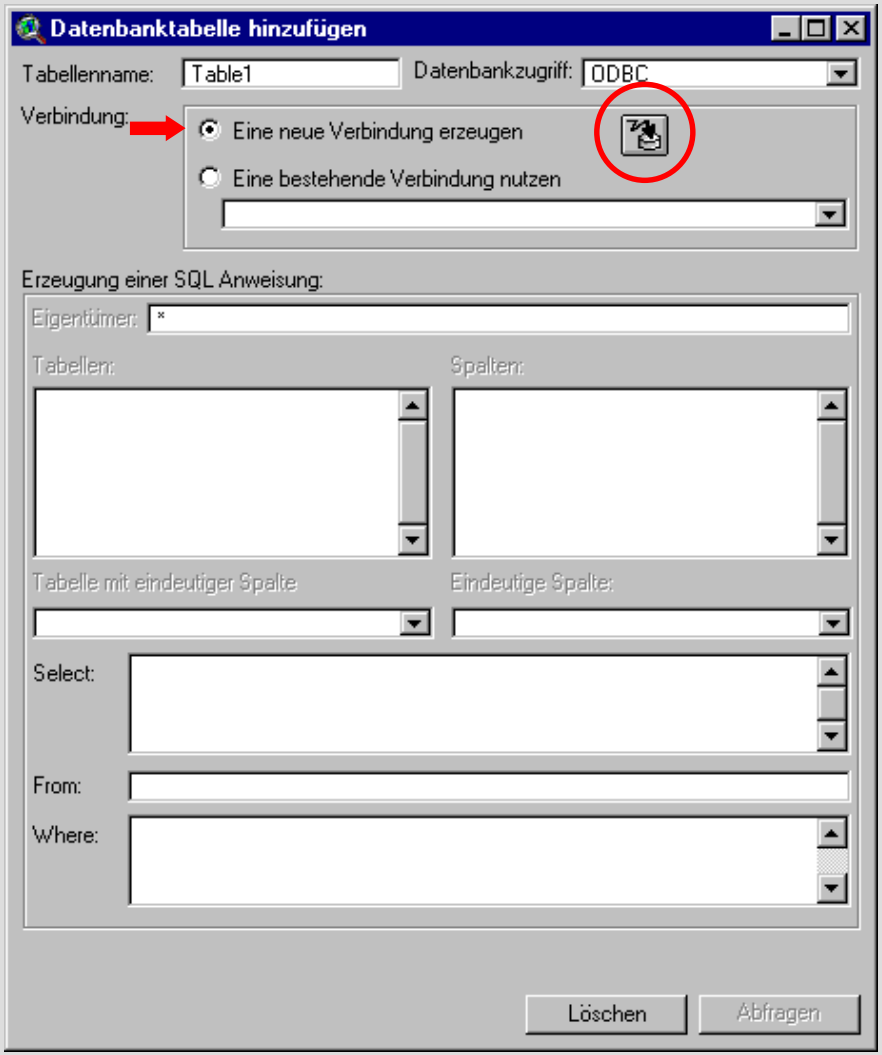


PAN Partnerschaft

SQL und Arcview: **Verbindungsaufbau über ODBC**

Das Hauptfenster für den Datenbankzugriff ist zunächst leer.

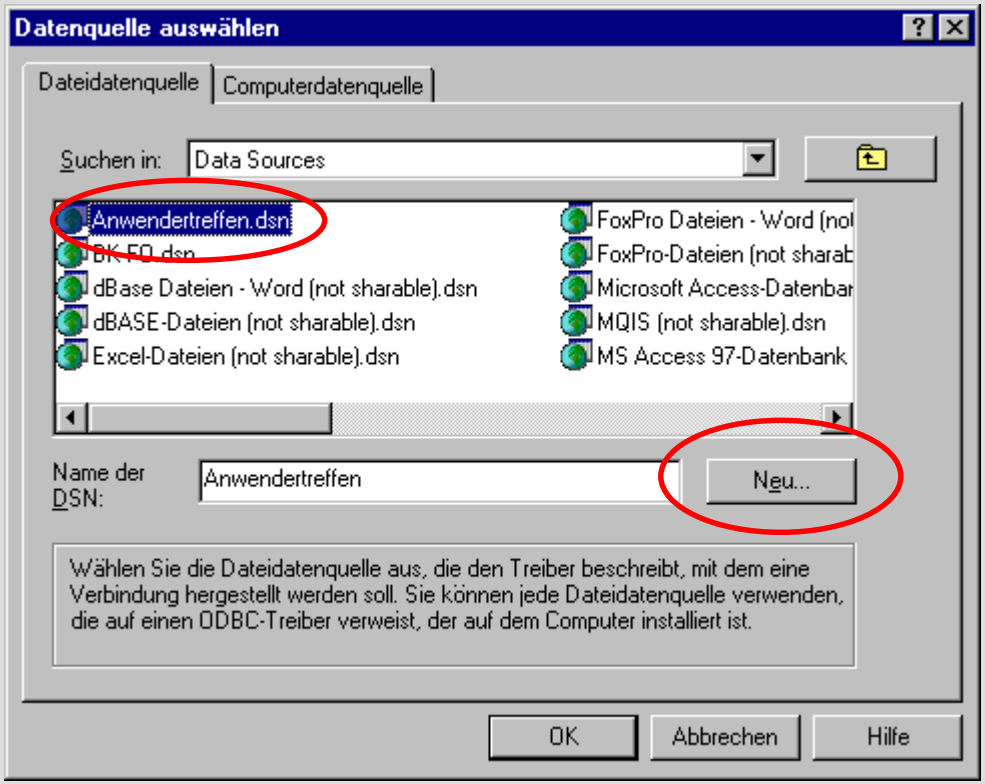
Als erstes muss eine Verbindung zu einer bestehenden Access-Datenbank hergestellt werden.



PAN Partnerschaft

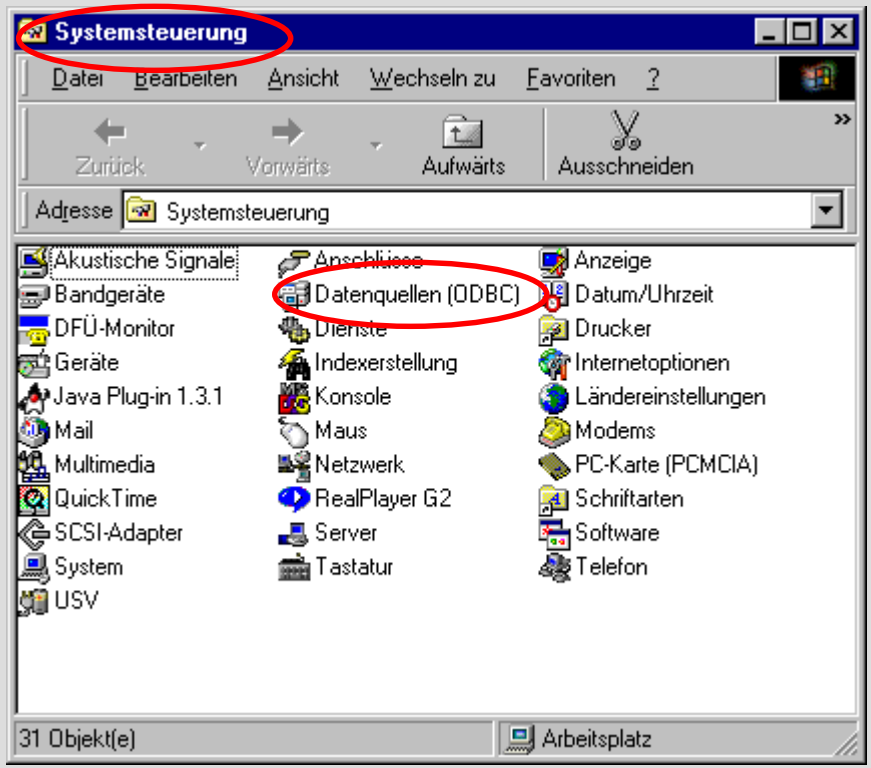
SQL und Arcview: **Verbindungsaufbau über ODBC**

Datenquellen können entweder neu definiert werden, oder es wird eine bereits bestehende Datenquelle verwendet.



SQL und Arcview: **Verbindungsaufbau über ODBC**

Unter Umständen kann es sinnvoller sein, die Datenquelle vor dem Start von ArcView zu definieren. Dies geschieht in der Systemsteuerung von Windows unter „**Datenquellen (ODBC)**“ (WIN 9x, WIN NT) bzw. unter „**Verwaltung - Datenquellen“** (WIN 2000).

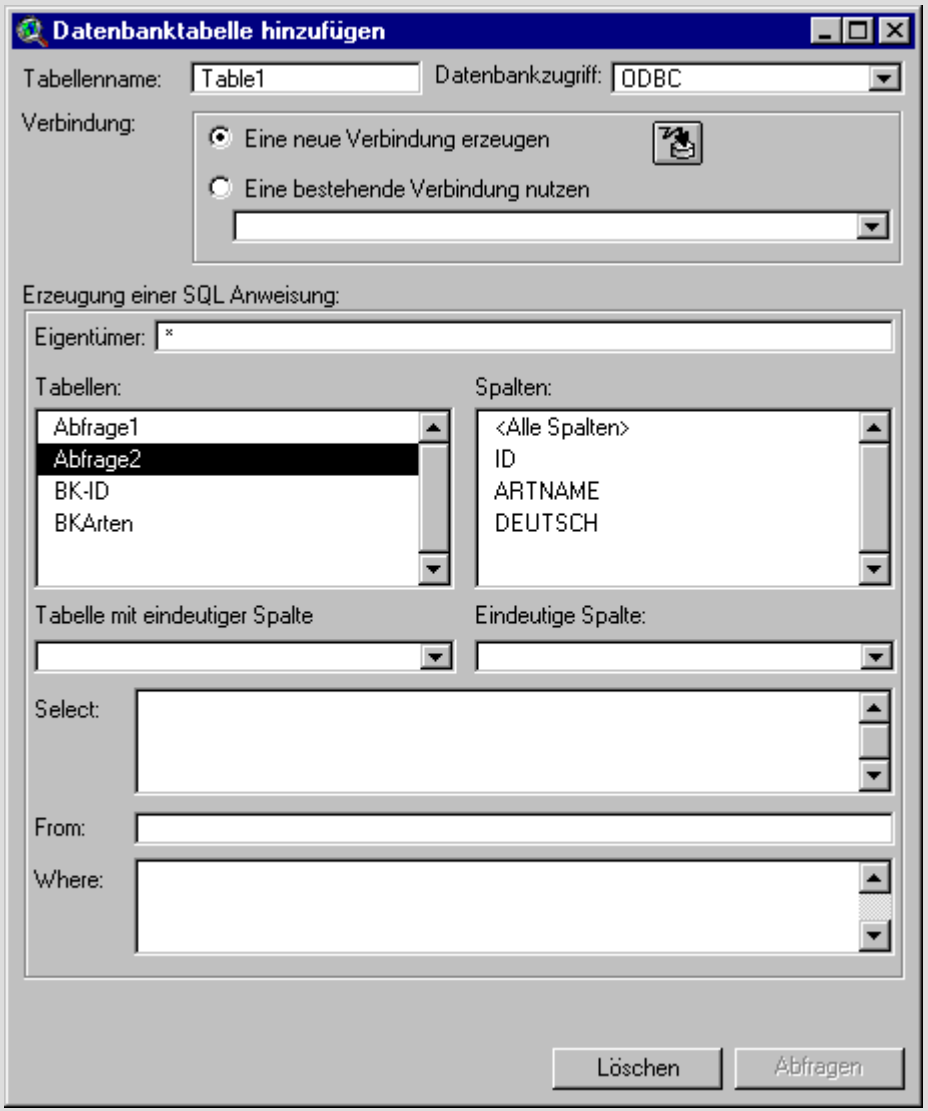


PAN Partnerschaft

SQL und Arcview: **Verfügbare Spalten mit Daten**

Nach Auswahl der Verbindung zur Datenbank sind im Fenster die verfügbaren Tabellen und Abfragen sichtbar.

Durch Auswahl einer Tabelle oder Abfrage werden auch die Spalten angezeigt, die in eine neue Tabelle eingebunden werden können.

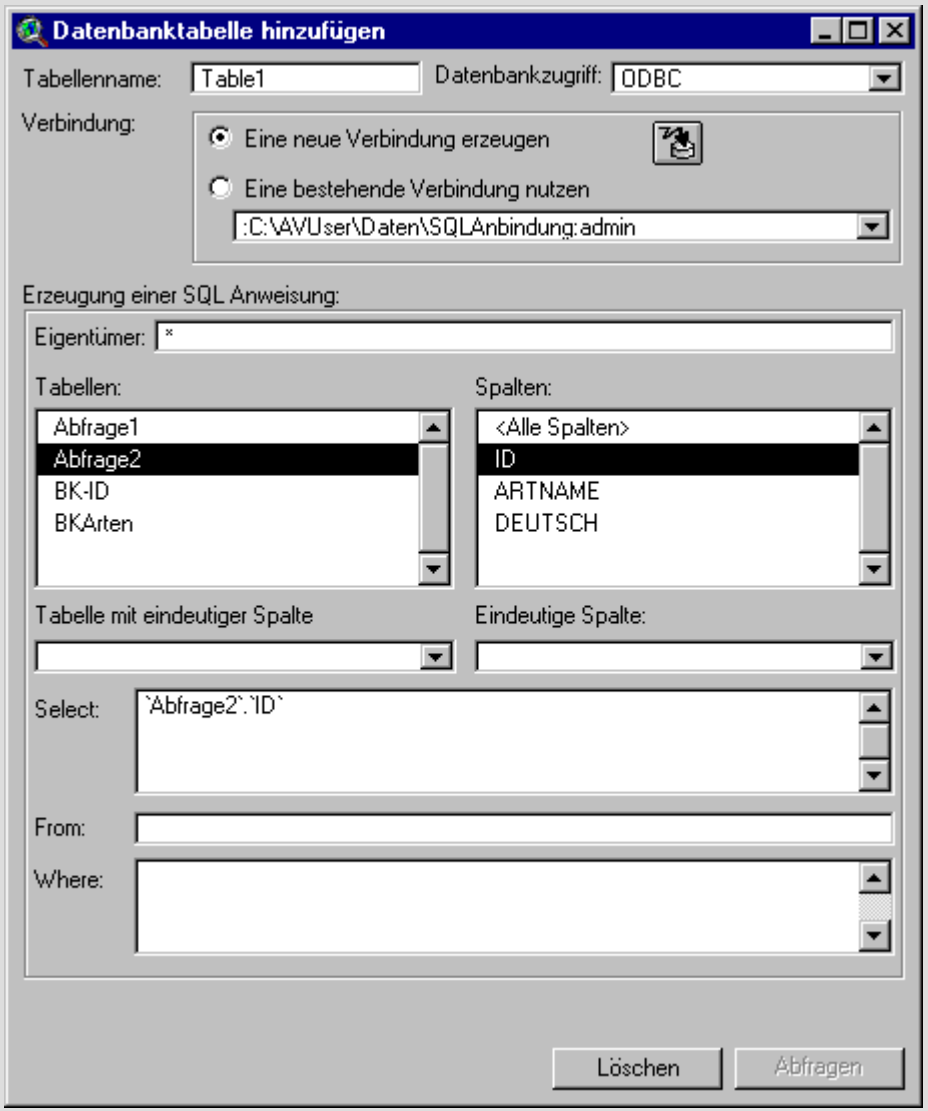


PAN Partnerschaft

SQL und Arcview: **Erzeugen der SQL-Abfrage**

Die SQL-Abfrage auf die Datenbank wird dann durch Doppelklicken auf die entsprechenden Spalten bzw. Tabellen oder Abfragen erstellt:

„SELECT“ wählt die Spalte(n) aus, die angezeigt werden sollen.

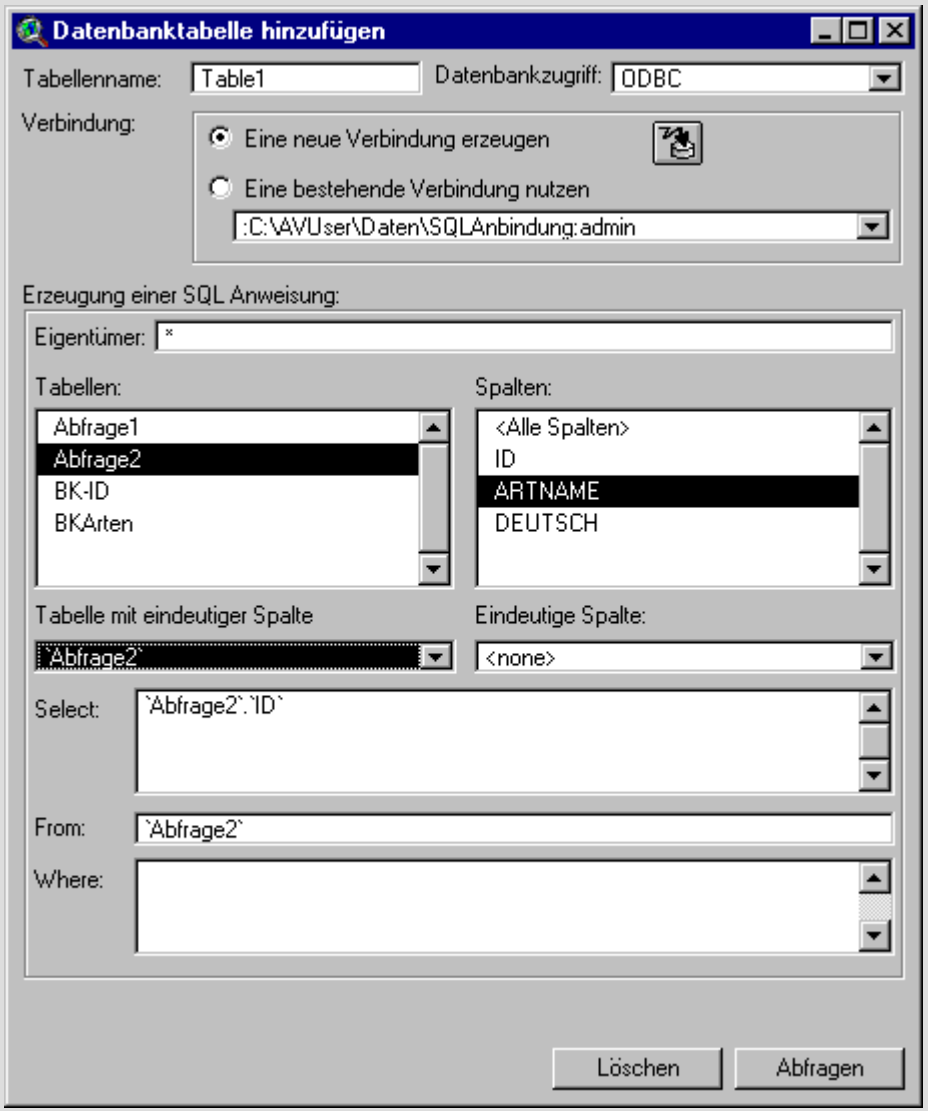


PAN Partnerschaft

SQL und Arcview: **Erzeugen der SQL-Abfrage**

Die SQL-Abfrage auf die Datenbank wird dann durch Doppelklicken auf die entsprechenden Spalten bzw. Tabellen oder Abfragen erstellt:

„FROM“ gibt den Namen der Tabellen und/oder Abfragen an, von der die Daten stammen.

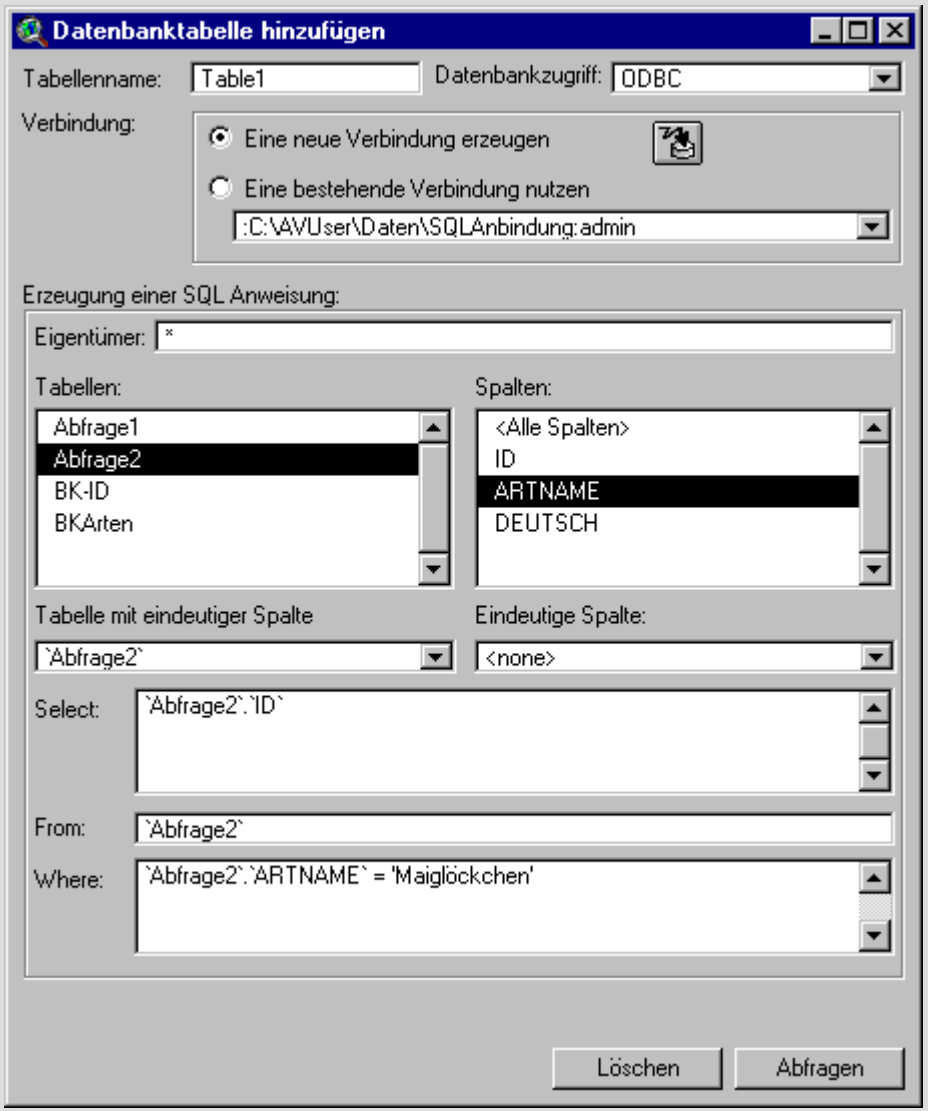


PAN Partnerschaft

SQL und Arcview: **Erzeugen der SQL-Abfrage**

Die SQL-Abfrage auf die Datenbank wird dann durch Doppelklicken auf die entsprechenden Spalten bzw. Tabellen oder Abfragen erstellt:

Unter „WHERE“ können eigene Kriterien formuliert werden.

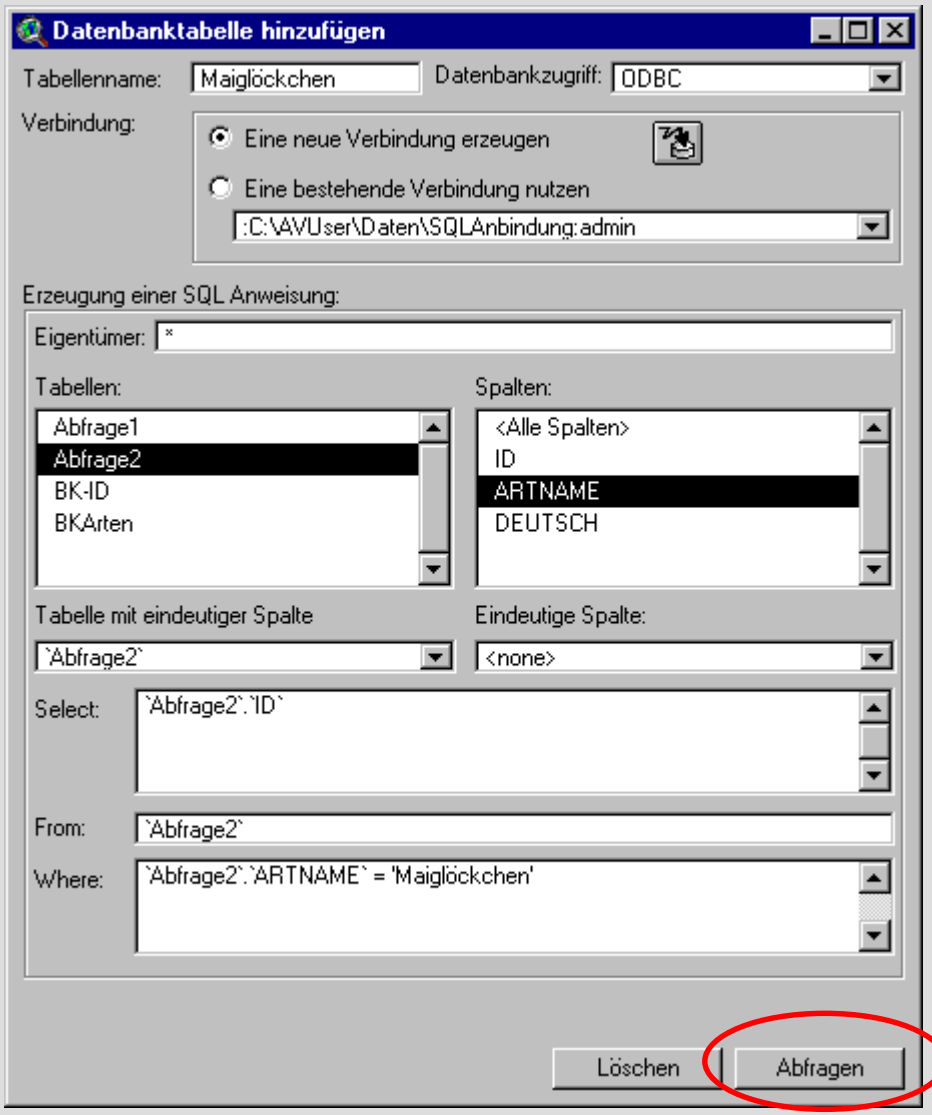


PAN Partnerschaft

SQL und Arcview: **Starten der Abfrage**

Bevor die Abfrage ausgeführt wird, sollte für die neu erstellte Tabelle von ein sinnvoller Name eingegeben werden.

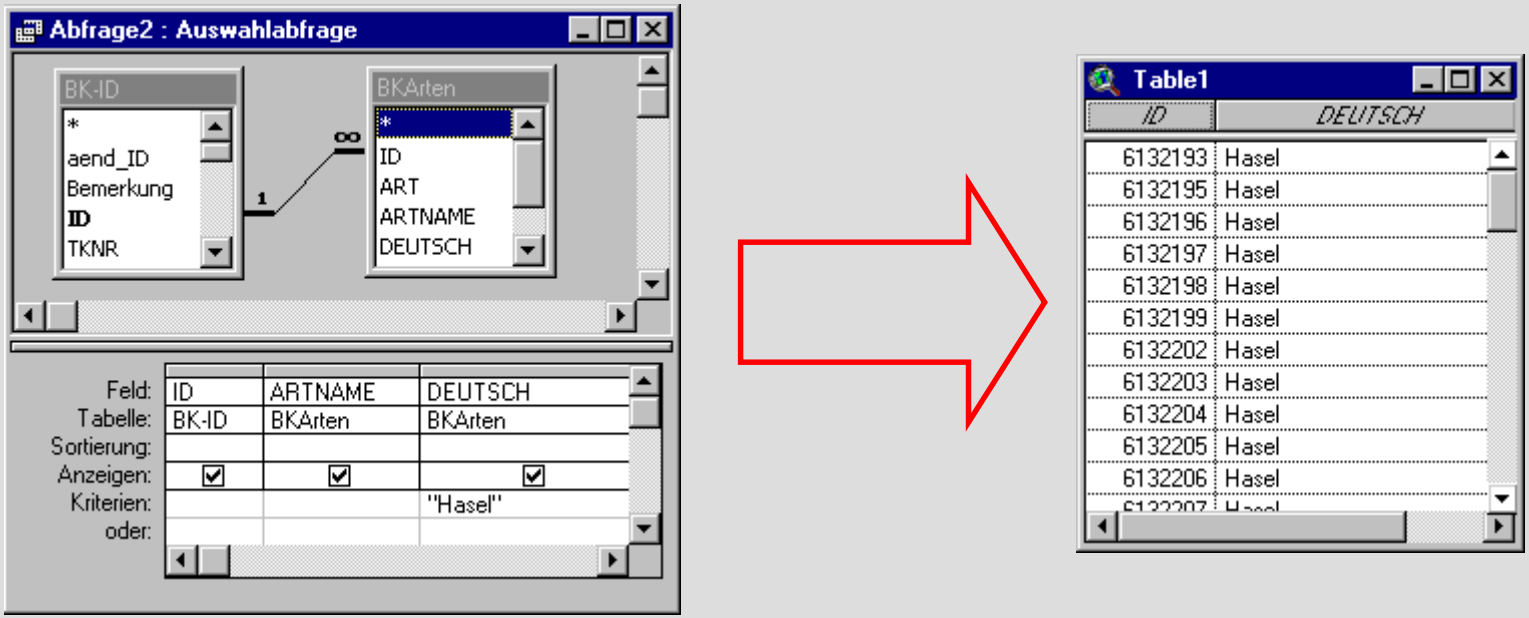
Gestartet wird dann mit der Schaltfläche „Abfragen“.



PAN Partnerschaft

SQL und Arcview: **dynamische Verbindung**

Das Ergebnis der Abfrage wird als neue Tabelle in ArcView angezeigt. Diese Tabelle ist **dynamisch**: wenn sich die Daten in der zugrundeliegenden Tabelle oder Abfrage in Access ändern, wird auch der Inhalt der Tabelle in ArcView aktualisiert.



PAN Partnerschaft

SQL und Arcview: **dynamische Verbindung**

Das Ergebnis der Abfrage wird als neue Tabelle in ArcView angezeigt. Diese Tabelle ist **dynamisch**: wenn sich die Daten in der zugrundeliegenden Tabelle oder Abfrage in Access ändern, wird auch der Inhalt der Tabelle in ArcView aktualisiert.

Darin liegt der große **Vorteil** der Einbindung von Datenbankdaten über SQL. Große Datenmengen können in Access gehalten werden, bei Bedarf werden Teilmengen in ArcView angezeigt, aber nicht fest gespeichert. Das ArcView-Projekt bleibt klein, die Anzahl an Objekten übersichtlich, die Reaktionszeit des Programms kurz.

Teil 3: SQL und Avenue-Programmierung

Teil 3: SQL und Avenue-Programmierung

In ArcView können alle über Menüs zugänglichen Funktionen des Programms auch mit Hilfe der Programmiersprache **Avenue** beeinflusst werden. Meist stehen dabei über die Programmierung mehr Möglichkeiten zur Verfügung, als es über die Menüs der Falls ist.

Auch im Falle der Anbindung von Datenbanken über ODBC und SQL ist die Programmierung über Avenue in vielen Fällen schneller und sinnvoller als die Verwendung der eben gezeigten Befehle.

SQL und Avenue: **Einstieg in die Programmierung**

Einfache Abfragen auf eine Datenbank lassen sich mit sehr wenigen Programmzeilen umsetzen. Um eine ähnliche Abfrage wie im gezeigten Beispiel zu erstellen, sind folgende Befehle notwendig:

SQL und Avenue: **Einstieg in die Programmierung**

Einfache Abfragen auf eine Datenbank lassen sich mit sehr wenigen Programmzeilen umsetzen. Um eine ähnliche Abfrage wie im gezeigten Beispiel zu erstellen, sind folgende Befehle notwendig:

verfügbare ODBC-Datenquellen anzeigen:

```
ODBCConnection.GetDatasources
```

Hier werden natürlich nur die ODBC-Verbindungen angezeigt, die vorher über die Systemsteuerung von Windows definiert wurden.

SQL und Avenue: **Einstieg in die Programmierung**

Einfache Abfragen auf eine Datenbank lassen sich mit sehr wenigen Programmzeilen umsetzen. Um eine ähnliche Abfrage wie im gezeigten Beispiel zu erstellen, sind folgende Befehle notwendig:

verfügbare ODBC-Datenquellen anzeigen:

```
ODBCConnection.GetDatasources
```

ODBC-Verbindung mit Datenbank herstellen:

```
theCON = ODBCConnection.Make("BK FO", "", "", "")
```

Für die Verbindung zu Access-Datenbanken ist beim Aufbau nur der Name der Datenquelle anzugeben (hier „BK FO“). Die restlichen drei Parameter in der Klammer werden nicht benötigt, hier genügt die Angabe eines „leeren“ Textes.

SQL und Avenue: **Einstieg in die Programmierung**

Einfache Abfragen auf eine Datenbank lassen sich mit sehr wenigen Programmzeilen umsetzen. Um eine ähnliche Abfrage wie im gezeigten Beispiel zu erstellen, sind folgende Befehle notwendig:

verfügbare ODBC-Datenquellen anzeigen:

```
ODBCConnection.GetDatasources
```

ODBC-Verbindung mit Datenbank herstellen:

```
theCON = ODBCConnection.Make("BK FO", "", "", "")
```

SQL-Abfrage als Text erzeugen:

```
theSQLString = "SELECT Abfrage2.* FROM Abfrage2"
```

Die Schreibweise der SQL-Abfrage entspricht genau derjenigen, die im Abfragefenster von Access selbst angezeigt werden kann, nur wird der Strichpunkt am Ende weggelassen.

SQL und Avenue: **Einstieg in die Programmierung**

Einfache Abfragen auf eine Datenbank lassen sich mit sehr wenigen Programmzeilen umsetzen. Um eine ähnliche Abfrage wie im gezeigten Beispiel zu erstellen, sind folgende Befehle notwendig:

verfügbare ODBC-Datenquellen anzeigen:

```
ODBCConnection.GetDatasources
```

ODBC-Verbindung mit Datenbank herstellen:

```
theCON = ODBCConnection.Make("BK FO", "", "", "")
```

SQL-Abfrage als Text erzeugen:

```
theSQLString = "SELECT Abfrage2.* FROM Abfrage2"
```

Abfrage aus Abfragetext erstellen:

```
theQDef = QueryDef.Make(theCON)
theQDef.SetSQL(theSQLString)
```

SQL und Avenue: **Einstieg in die Programmierung**

Einfache Abfragen auf eine Datenbank lassen sich mit sehr wenigen Programmzeilen umsetzen. Um eine ähnliche Abfrage wie im gezeigten Beispiel zu erstellen, sind folgende Befehle notwendig:

verfügbare ODBC-Datenquellen anzeigen:

```
ODBCConnection.GetDatasources
```

ODBC-Verbindung mit Datenbank herstellen:

```
theCON = ODBCConnection.Make("BK FO", "", "", "")
```

SQL-Abfrage als Text erzeugen:

```
theSQLString = "SELECT Abfrage2.* FROM Abfrage2"
```

Abfrage aus Abfragetext erstellen:

```
theQDef = QueryDef.Make(theCON)
theQDef.SetSQL(theSQLString)
```

Datenbanktabelle durch Senden der Abfrage erstellen:

```
theDBTabelle = DBTable.Make(theQDef)
```

SQL und Avenue: **Aktualisierung von Daten**

Wenn eine Verbindung zur Datenbank hergestellt ist und Daten in ArcView eingebunden wurden, können durch Änderung der Abfrage diese Daten aktualisiert werden, ohne dass eine neue Tabelle erstellt werden muss:

SQL und Avenue: **Aktualisierung von Daten**

Wenn eine Verbindung zur Datenbank hergestellt ist und Daten in ArcView eingebunden wurden, können durch Änderung der Abfrage diese Daten aktualisiert werden, ohne dass eine neue Tabelle erstellt werden muss:

SQL-Abfrage neu definieren:

```
theSQLString =
"SELECT Abfrage2.* FROM Abfrage2 WHERE Abfrage2.ID = 6232262"
```

Die Abfrage entspricht dem ersten Beispiel, nur wird über den „WHERE"-Ausdruck eine Bedingung angegeben. In diesem Fall sollen alle Datensätze mit der ID 6232 262 ausgegeben werden.

SQL und Avenue: **Aktualisierung von Daten**

Wenn eine Verbindung zur Datenbank hergestellt ist und Daten in ArcView eingebunden wurden, können durch Änderung der Abfrage diese Daten aktualisiert werden, ohne dass eine neue Tabelle erstellt werden muss:

SQL-Abfrage neu definieren:

```
theSQLString =
"SELECT Abfrage2.* FROM Abfrage2 WHERE Abfrage2.ID = 6232262"
```

bestehende Datenbanktabelle finden:

```
theDBTabelle = av.FindDoc("Table1")
```

Falls die bestehende Datenbanktabelle umbenannt wurde, muss hier natürlich statt „Table1“ der entsprechende Name angegeben werden.

SQL und Avenue: **Aktualisierung von Daten**

Wenn eine Verbindung zur Datenbank hergestellt ist und Daten in ArcView eingebunden wurden, können durch Änderung der Abfrage diese Daten aktualisiert werden, ohne dass eine neue Tabelle erstellt werden muss:

SQL-Abfrage neu definieren:

```
theSQLString =
"SELECT Abfrage2.* FROM Abfrage2 WHERE Abfrage2.ID = 6232262"
```

bestehende Datenbanktabelle finden:

```
theDBTabelle = av.FindDoc("Table1")
```

neue Abfrage aus Abfragetext erstellen:

```
theQDef = theDBTabelle.GetQueryDef
theQDef.SetSQL(theSQLString)
```

SQL und Avenue: **Aktualisierung von Daten**

Wenn eine Verbindung zur Datenbank hergestellt ist und Daten in ArcView eingebunden wurden, können durch Änderung der Abfrage diese Daten aktualisiert werden, ohne dass eine neue Tabelle erstellt werden muss:

SQL-Abfrage neu definieren:

```
theSQLString =
"SELECT Abfrage2.* FROM Abfrage2 WHERE Abfrage2.ID = 6232262"
```

bestehende Datenbanktabelle finden:

```
theDBTabelle = av.FindDoc("Table1")
```

neue Abfrage aus Abfragetext erstellen:

```
theQDef = theDBTabelle.GetQueryDef
theQDef.SetSQL(theSQLString)
```

Datenbanktabelle aktualisieren:

```
theDBTabelle.Refresh
```

SQL und Avenue: **weitere Schritte**

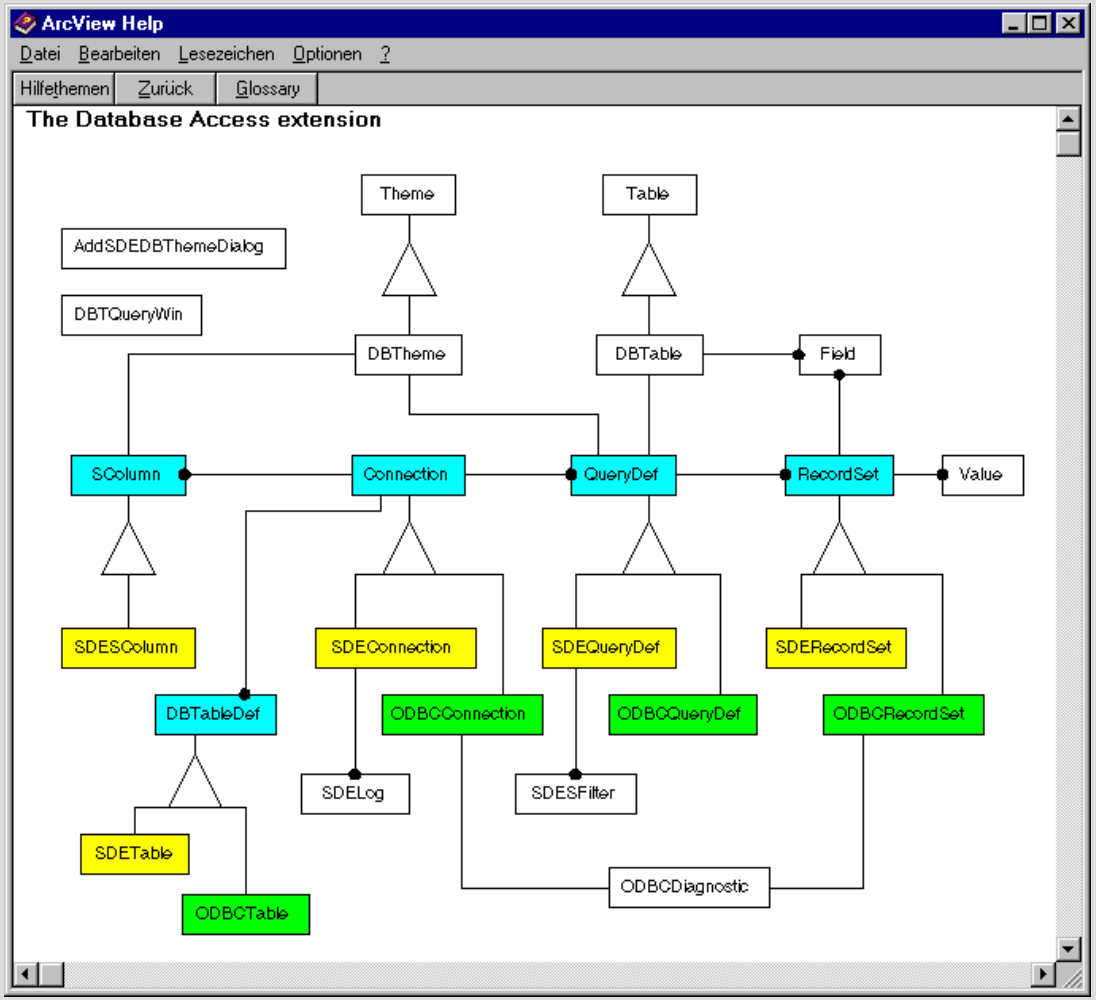
ArcView bietet noch zahlreiche weitere Möglichkeiten, über Programmierung mit Avenue mit Datenbanken zusammenzuarbeiten. Insbesondere können über den SQL-Zugriff auch die Daten in der Datenbank selbst verändert werden.

Wenn als Datenbank allerdings Access verwendet wird, sind diese Möglichkeiten für die meisten Anwendungen eher uninteressant. Datenänderungen können nämlich in Access selbst wesentlich bequemer vorgenommen werden.

SQL und Avenue: **weitere Schritte**

Weitere Objekte und Methoden zur Programmierung der Einbindung von Datenbanken finden sich in der ArcView-Hilfe zum Thema „Database Access“.

Hier sind auch zahlreiche Beispielskripte vorhanden.



PAN Partnerschaft

Teil 4: DDE – Datenaustausch in beiden Richtungen

Teil 4: DDE – Datenaustausch in beiden Richtungen

SQL ist eine Abfragesprache, die von fast allen Datenbanksystemen verstanden wird. Dies gilt nicht nur für Windows, sondern z. B. auch für Datenbanken unter UNIX.

Diese Allgemeingültigkeit bringt für die Einbindung von Datenbanken in ArcView zwei wesentliche Nachteile mit sich:

1. Der Datenaustausch kann nur von ArcView aus erfolgen. Eine Access-Datenbank hat keine Möglichkeit, Daten an ArcView zu senden.
2. Viele der „komfortablen" Eigenschaften von Windows-Programmen wie Access (z. B. Formulare) können nicht eingebunden werden.

DDE: **Definition und Grundlagen**

Für den Datenaustausch zwischen ArcView und Windows-Programmen stellt das Windows-Betriebssystem einen Mechanismus zur Verfügung, der **DDE** genannt wird. Dies steht für „dynamic data exchange“ oder „dynamischer Datenaustausch“.

Unter DDE kommunizieren zwei Anwendungen, die gleichzeitig auf dem selben Computer ausgeführt werden. Dabei nimmt eine Anwendung die Rolle des Datensenders an (server), die andere die Rolle des Datenempfängers (client). Beide Anwendungen können wechselseitig jede dieser Rollen annehmen.

Mit DDE können die beiden Anwendungen kontinuierlich Daten austauschen. Dabei geht es nicht nur um „Daten“ im Sinne von Zahlen oder Text, sondern auch um Programmierbefehle wie Avenue-Skripten oder Makros.

DDE: **Grundbegriffe**

Unter dem DDE-Mechanismus sind drei Methoden der Kommunikation möglich, die in den jeweiligen Programmen verschieden umgesetzt werden:

Execute: die Client-Anwendung fordert die Server-Anwendung auf, eine Funktion ablaufen zu lassen.

Beispiel: Access (als Client) schickt einen Befehl an ArcView (als Server), der in ArcView ein Skript ausführt.

DDE: **Grundbegriffe**

Unter dem DDE-Mechanismus sind drei Methoden der Kommunikation möglich, die in den jeweiligen Programmen verschieden umgesetzt werden:

Execute: die Client-Anwendung fordert die Server-Anwendung auf, eine Funktion ablaufen zu lassen.

Request: die Client-Anwendung fordert die Server-Anwendung auf, einen Wert zurückzugeben.

Beispiel: ArcView (als Client) schickt an Access (als Server) die Aufforderung, einen bestimmten Wert aus einer Tabelle zurückzugeben, der dann in ArcView weiterverarbeitet wird.

DDE: **Grundbegriffe**

Unter dem DDE-Mechanismus sind drei Methoden der Kommunikation möglich, die in den jeweiligen Programmen verschieden umgesetzt werden:

Execute: die Client-Anwendung fordert die Server-Anwendung auf, eine Funktion ablaufen zu lassen.

Request: die Client-Anwendung fordert die Server-Anwendung auf, einen Wert zurückzugeben.

Poke: die Client-Anwendung sendet Daten an die Server-Anwendung und fordert diese auf, die Daten zu verarbeiten.

Beispiel: ArcView (als Client) sendet die Flächengröße eines Polygons an Access (als Server) und fordert Access auf, diesen Wert in eine Tabelle einzutragen.

DDE: **Aufbau der Kommunikation**

Damit die beteiligten Programme kommunizieren können, muss zunächst von der Client-Anwendung aus die Kommunikation gestartet werden.

In ArcView - Avenue erfolgt dies mit dem Befehl:

```
DDEClient.Make (ServerName, TopicName)
```

Dabei ist „ServerName“ der Name des Programms, „TopicName“ das Thema. Was ein Thema im Sinne von DDE ist, wird von jeder Anwendung unterschiedlich interpretiert. In Access ist es einfach der vollständige Pfad zur Datenbank, die verbunden werden soll:

```
DDEClient.Make("MSAccess", "c:\temp\daten.mdb")
```

DDE: **Aufbau der Kommunikation**

Hier als Beispiel ein vollständiges Avenue-Skript. Falls der Aufbau fehlschlägt, weil Access noch nicht geöffnet war, wird das Programm gestartet und die Anweisung dann wiederholt.

```
theClient = DDEClient.Make("MSAccess", "c:\temp\daten.mdb")

if (theClient.HasError) then
   System.Execute("C:\Programme\Microsoft Office\
   Office\MSACCESS.EXE "+"c:\temp\daten.mdb")
end

theClient = DDEClient.Make("MSAccess", "c:\temp\daten.mdb")
```

DDE: **Aufbau der Kommunikation**

Analog kann die Kommunikation auch von Access aus gestartet werden. Hier müssen die Befehle mit der Programmiersprache VBA erstellt werden. Der entsprechende Befehl lautet:

```
DDEInitiate(Anwendung, Thema)
```

Auch hier ist unter „Anwendung“ der Name des angesprochenen Programms einzusetzen. Bei der Kommunikation mit ArcView gibt es in jedem Fall nur ein Thema, das als „System“ bezeichnet wird. Die konkrete Anweisung lautet also:

```
DDEInitiate("Arcview", "System")
```

DDE: **von ArcView aus ein Access-Makro starten**

Als Beispiel für eine DDE-Kommunikation soll von ArcView aus in Access ein Formular geöffnet werden. Dazu wird die Execute-Methode verwendet.

Die Execute-Methode wird in der Avenue-Programmierung folgendermaßen verwendet:

```
DDEClient.Execute (aTask)
```

Was ein Task ist, hängt von der jeweiligen Server-Anwendung ab. In Access kann jeder Befehl, der als Makro verwendet wird, auch als Task angenommen werden.

DDE: **von ArcView aus ein Access-Makro starten**

Hier das vollständige Beispiel als ein Avenue-Skript. Zunächst wird die Kommunkation wie schon gezeigt gestartet.

Anschließend wird der Execute-Befehl verwendet, dabei wird in eckigen Klammern die Makro-Anweisung für das Öffnen des Formulars und der Formularname an Access gesandt.

Schließlich soll im Formular noch ein bestimmter Datensatz angezeigt werden. Dazu wird ein Filter gesetzt - wieder über die entsprechende Makro-Anweisung, und der gewünschte Wert weitergegeben.

```
theClient = DDEClient.Make("MSAccess", "c:\temp\daten.mdb")
theClient.Execute("[OpenForm , ""Formular1""]")
theClient.Execute("[ApplyFilter , "ID=6232262"]")
```

DDE: **von Access aus ein Objekt in ArcView auswählen**

Als zweites Beispiel - ebenfalls für die Execute-Methode - soll jetzt von Access aus eine bestimmte Fläche in ArcView ausgewählt werden.

Die Execute-Methode wird in VBA folgendermaßen verwendet:

```
DDEExecute Kanalnummer, Befehl
```

Dabei steht, ganz ähnlich wie in ArcView, die Kanalnummer für die eben aufgebaute Kommunikation, und „Befehl“ für ein Anweisung, die die jeweils andere Anwendung versteht. Bezüglich ArcView kann „Befehl“ jede in Avenue geschriebene Anweisung sein.

DDE: **von Access aus ein Objekt in ArcView auswählen**

Hier das vollständige Beispiel als VBA-Funktion. Zunächst wird wieder die Kommunikation gestartet.

Anschließend wird ein Befehl als Text erstellt - in diesem Fall die Anweisung, ein bestimmtes Skript auszuführen (in Avenue: „av.run“). Dabei wird wieder ein Parameter übergeben.

Schließlich wird die Execute-Methode mit dem erstellten Befehl ausgeführt.

```
Kanal = DDEInitiate("Arcview", "System")

AvenueBefehl =
"av.run(""DDE.SelectShapes"", ""6232262"")"

DDEExecute Kanal, AvenueBefehl
```

4. ArcView Anwendertreffen

Datenbankanbindung mit ArcView

Eine PDF-Fassung dieses Vortrags und zwei ArcView-Projekte mit allen Beispielskripts können von unserer Homepage heruntergeladen werden:

www.pan-partnerschaft.de

PAN Partnerschaft