Der dBASE-ODBC-Treiber

Hinweis Dieser Treiber ist nur für die Verwendung mit Lotus-Anwendungen lizensiert.

Einführung

Der Lotus-ODBC-dBASE-Treiber unterstützt:

- nur Lotus-Anwendungen
- Dateien in den Formaten dBASE III und IV
- Clipper-Dateien
- FoxPro- und FoxBASE-Dateien

Der dBASE-Treiber führt die SQL-Anweisungen direkt auf dBASE-, Clipper-, FoxPro- und FoxBASE-Dateien aus. Sie müssen keines der dBASE-Produkte besitzen, um auf diese Dateien zuzugreifen.

Der dBASE-Treiber bietet folgende Funktionen:

- Transaktionsunterstützung für dBASE-kompatible Dateien
- Eingesetzte Trennebene beim Lesen der Daten
- Dateisperre.

dBASE-Datenquellen konfigurieren

Hinweis Dieser Treiber ist nur für die Verwendung mit Lotus-Anwendungen lizensiert.

Eine dBASE-Datenbank wird an ihrer Datenquellendefinition erkannt. Alle Datenquellendefinitionen enthalten:

- einen Datenquellennamen, der die Verbindung eindeutig kennzeichnet. Beispiele: "Meine Kontendatenbank" oder "dBASE-Dateien in Verzeichnis C:\DBF."
- das Verzeichnis und den Pfad der Datenbank, zum Beispiel: C:\DBF

Lotus-Anwendungen erstellen ODBC-Datenquellen automatisch, wenn Sie ein Verzeichnis und einen Pfad angeben. Sie können eine dBASE-Datenquelle manuell mit dem ODBC-Administrator-Tool konfigurieren und die Definition der Optionen im Dialogfeld ODBC dBASE-Setup vornehmen. Das Setup-Dialogfeld ermöglicht es Ihnen, zusätzliche Informationen über die Datenverbindung aufzunehmen.

Zur Konfiguration einer dBASE-Datenquelle gehen Sie wie folgt vor:

1. Doppelklicken Sie in der Systemsteuerung von Windows auf das ODBC-Symbol.

   Es erscheint eine Liste von Datenquellen.

2. Klicken Sie auf Hinzufügen, wenn Sie eine neue Datenquelle konfigurieren möchten.

   Es erscheint eine Liste der installierten Treiber.

3. Wählen Sie Lotus Q+E dBASE, und klicken Sie auf OK.
4. Wenn Sie eine vorhandene Datenquelle konfigurieren möchten, wählen Sie den Namen der Datenquelle und klicken dann auf Einrichten.
5. Es erscheint das Setup-Dialogfeld. Geben Sie im Dialogfeld ODBC dBASE-Setup die Information zum Einrichten der Datenquelle.
6. Klicken Sie auf OK, um das Setup-Dialogfeld zu verlassen. Der Treiber schreibt diese Werte in die ODBC.INI-Datei. Diese Werte sind nun jedesmal, wenn Sie eine Verbindung zu der Datenquelle herstellen, die Vorgabewerte. Sie können die in der Datei ODBC.INI gespeicherten Vorgabewerte ändern, indem Sie Ihre Datenquelle erneut konfigurieren.

dBASE-IV-Treiberkommunikation und Sicherheit

Berücksichtigen Sie die folgenden, die Sicherheit und Kommunikation betreffenden Punkte, während Sie mit dem dBASE-IV-Treiber arbeiten:

- Der dBASE-IV-Treiber arbeitet direkt mit der dBASE-IV-Tabelle. Die Benutzer benötigen kein dBASE IV, um auf die Tabellen zuzugreifen.
- Es gibt kein vorgegebenes Schutzschema für Datenbanken. Der dBASE-IV-Treiber unterstützt keine Kennwörter oder Benutzer-IDs. Damit Sie eine dBASE-IV-Tabelle verwenden können, müssen Sie ohne Verwendung eines Kennworts oder einer Benutzer-ID auf die Tabelle zugreifen können. Der dBASE-IV-Treiber unterstützt keine verschlüsselten Dateien.
- Löscht ein Benutzer eine Tabelle, löscht der dBASE-IV-Treiber die .MDX-Dateien, die mit dem gleichen Namen wie die gelöschte Tabelle benannt wurden, aber keine zugeordneten Masken-, Report- oder Indexdateien.
- Bei der gemeinsamen Nutzung von Dateien auf einem Netzwerk mit der vorgegebenen Einstellung Locking=File können Sie nur dann in eine Tabelle schreiben, wenn gerade niemand anders die Tabelle modifiziert.
- Benutzer, die auf eine dBASE-IV-Tabelle nur einen Lese-Zugriff haben, können die Daten in der Tabelle lesen, aber nicht in die Tabelle schreiben, Datensätze aus der Tabelle löschen oder die Tabelle löschen.
- Sie kontrollieren den Zugriff der Benutzer auf die dBASE-IV-Tabellen mit den Standard-Betriebssystembefehlen oder Netzwerksicherheitsmechanismen.

Unterstützte dBASE-IV-Zeichensätze

Der dBASE-IV-Treiber unterstützt die folgenden Zeichensätze: USA (Zeichensatztabelle 437), Mehrsprachig (Zeichensatztabelle 850), Portugiesisch (Zeichensatztabelle 860), Französisch-Kanadisch (Zeichensatztabelle 863) und Norwegisch (Zeichensatztabelle 865). Der US-Zeichensatz ist vorgegeben.

dBASE-IV-Datentypen

Die folgende Liste definiert die dBASE-IV-Datentypen und zeigt, wie diese auf den Standard-ODBC-Datentypen abgebildet werden. Die folgenden dBASE-IV-Datentypen können in einer CREATE TABLE-Anweisung verwendet werden.

dBASE-IV-Datentyp: Logical (Logisch)

ODBC-Datentyp: SQL_BIT

Enthält die Informationen wahr/falsch oder ja/nein. Die dBASE-Datei enthält als mögliche Werte die Buchstaben T, F, Y oder N. Auf der Anwendungs- und ODBC-API-Ebene sind die möglichen Werte 1 oder 0.

dBASE-IV-Datentyp: Char (Zeichen)

ODBC-Datentyp: SQL_CHAR

Die Werte können Buchstaben, Ziffern oder jedes der Satzzeichen auf Ihrer Tastatur enthalten. Es ist ein Parameter für die Länge erforderlich, der die maximale Länge eines Zeichenwertes angibt, der gespeichert werden kann. Die maximale Länge beträgt 254 Zeichen. Ein Beispiel: CHAR(12).

dBASE-IV-Datentyp: Memo

ODBC-Datentyp: SQL_LONGVARCHAR

Die Werte enthalten lange, mehrzeilige Textdaten.

dBASE-IV-Datentyp: Float (Gleit)

ODBC-Datentyp: SQL_DECIMAL

Die Werte werden wie bei dem Datentyp Numeric (Numerisch) abgespeichert. Float hat die gleichen Parameter wie Numeric. Zum Beispiel sind FLOAT(10,2) und NUMERIC(10,2) gleich.

dBASE-IV-Datentyp: Numeric (Numerisch)

ODBC-Datentyp: SQL_DECIMAL

Werte dürfen nur Ziffern, ein Dezimaltrennzeichen und ein führendes Minuszeichen enthalten. Mit zwei Parametern wird die maximale Anzahl der Stellen in der Zahl und, optional, die Anzahl der Stellen rechts vom Dezimaltrennzeichen angegeben. Die Gesamtzahl der Stellen ist auf 19 begrenzt. Beispiel: NUMERIC(10,2) deklariert eine 10-stellige Zahl mit 8 Vorkomma- und 2 Nachkommastellen. NUMERIC(8) ist gleichbedeutend mit NUMERIC(8,0) und deklariert eine 8-stellige Zahl ohne Nachkommastellen.

dBASE-IV-Datentyp: Date (Datum)

ODBC-Datentyp: SQL_DATE

Die Werte enthalten Datumswerte. Die Tageszeit ist nicht enthalten. Ein Beispiel: DATE.

dBASE-III- und Clipper-Datentypen

Die folgende Liste definiert die dBASE-III- und Clipper-Datentypen und zeigt, wie diese auf den Standard-ODBC-Datentypen abgebildet werden. Die folgenden Datentypen können in einer CREATE TABLE-Anweisung verwendet werden.

dBASE-III- oder Clipper-Datentyp: Logical (Logisch)

ODBC-Datentyp: SQL_BIT

Enthält die Informationen wahr/falsch oder ja/nein. Die dBASE-Datei enthält als mögliche Werte die Buchstaben T, F, Y oder N. Auf der Anwendungs- und ODBC-API-Ebene sind die möglichen Werte 1 oder 0.

dBASE-III- oder Clipper-Datentyp: Char (Zeichen)

ODBC-Datentyp: SQL_CHAR

Die Werte können Buchstaben, Ziffern oder jedes der Satzzeichen auf Ihrer Tastatur enthalten. Es ist ein Parameter für die Länge erforderlich, der die maximale Länge eines Zeichenwertes angibt, der gespeichert werden kann. Die maximale Länge für dBASE III beträgt 254 Zeichen und für Clipper 1024 Zeichen. Ein Beispiel: CHAR(12).

dBASE-III- oder Clipper-Datentyp: Memo

ODBC-Datentyp: SQL_LONGVARCHAR

Die Werte enthalten lange, mehrzeilige Textdaten.

dBASE-III- oder Clipper-Datentyp: Numeric (Numerisch)

ODBC-Datentyp: SQL_DECIMAL

Werte dürfen nur Ziffern, ein Dezimaltrennzeichen und ein führendes Minuszeichen enthalten. Mit zwei Parametern wird die maximale Anzahl der Stellen in der Zahl und, optional, die Anzahl der Stellen rechts vom Dezimaltrennzeichen angegeben. Die Gesamtzahl der Stellen ist auf 19 begrenzt. Beispiel: NUMERIC(10,2) deklariert eine 10-stellige Zahl mit 8 Vorkomma- und 2 Nachkommastellen. NUMERIC(8) ist gleichbedeutend mit NUMERIC(8,0) und deklariert eine 8-stellige Zahl ohne Nachkommastellen.

dBASE-III- oder Clipper-Datentyp: Date (Datum)

ODBC-Datentyp: SQL_DATE

Die Werte enthalten Datumswerte. Die Tageszeit ist nicht enthalten. Ein Beispiel: DATE.

FoxPro- und FoxBASE-Datentypen

Die folgende Liste definiert die FoxPro- und FoxBASE-Datentypen und zeigt, wie diese auf den Standard-ODBC-Datentypen abgebildet werden. Die folgenden Datentypen können in einer CREATE TABLE-Anweisung verwendet werden.

FoxPro- oder FoxBASE-Datentyp: Logical (Logisch)

ODBC-Datentyp: SQL_BIT

Enthält die Informationen wahr/falsch oder ja/nein. Die dBASE-Datei enthält als mögliche Werte die Buchstaben T, F, Y oder N. Auf der Anwendungs- und ODBC-API-Ebene sind die möglichen Werte 1 oder 0.

FoxPro- oder FoxBASE-Datentyp: Char (Zeichen)

ODBC-Datentyp: SQL_CHAR

Die Werte können Buchstaben, Ziffern oder jedes der Satzzeichen auf Ihrer Tastatur enthalten. Es ist ein Parameter für die Länge erforderlich, der die maximale Länge eines Zeichenwertes angibt, der gespeichert werden kann. Die maximale Länge beträgt 254 Zeichen.

FoxPro- oder FoxBASE-Datentyp: Memo

ODBC-Datentyp: SQL_LONGVARCHAR

Die Werte enthalten lange, mehrzeilige Textdaten.

FoxPro- oder FoxBASE-Datentyp: Numeric (Numerisch)

ODBC-Datentyp: SQL_DECIMAL

Werte dürfen nur Ziffern, ein Dezimaltrennzeichen und ein führendes Minuszeichen enthalten. Mit zwei Parametern wird die maximale Anzahl der Stellen in der Zahl und, optional, die Anzahl der Stellen rechts vom Dezimaltrennzeichen angegeben. Die Gesamtzahl der Stellen ist auf 19 begrenzt. Beispiel: NUMERIC(10,2) deklariert eine 10-stellige Zahl mit 8 Vorkomma- und 2 Nachkommastellen. NUMERIC(8) ist gleichbedeutend mit NUMERIC(8,0) und deklariert eine 8-stellige Zahl ohne Nachkommastellen.

FoxPro- oder FoxBASE-Datentyp: Date (Datum)

ODBC-Datentyp: SQL_DATE

Die Werte enthalten Datumswerte. Die Tageszeit ist nicht enthalten. Ein Beispiel: DATE.

FoxPro- oder FoxBASE-Datentyp: Picture (Bild)

ODBC-Datentyp: SQL_LONGVARBINARY

Werte enthalten grafische Bilder.

FoxPro-2.5-Datentyp: General

ODBC-Datentyp: SQL_LONGVARBINARY

Werte enthalten Binärdaten wie OLE-Objekte. Berücksichtigen Sie, daß grafische Bilddaten nicht unter diesem Datentyp gespeichert werden. Dieser Datentyp ist nur für FoxPro 2.5 zulässig.

ROWID-Pseudospalte

Jeder Datensatz enthält ein spezielles Feld mit der Bezeichnung ROWID. Dieses Feld enthält eine eindeutige Zahl, welche die Datensatzfolge in der Datenbank enthält. Zum Beispiel hat eine Tabelle, die 50 Datensätze enthält, ROWID-Werte von 1 bis 50 (falls keine Datensätze als gelöscht markiert sind). Sie können ROWID in WHERE- und SELECT-Klauseln verwenden.

Eine einzelne Zeile läßt sich am schnellsten durch Verwendung einer WHERE-Klausel mit der ROWID aktualisieren. Sie können jedoch die ROWID-Spalte nicht aktualisieren.

Optionen bei der FROM-Klausel

Die FROM-Klausel bietet eine Anzahl Optionen, die Sie mit Tabellennamen verwenden können. Diese Optionen stellen die Kompatibilität zu früheren Versionen von Q+E-Produkten her und ermöglichen es Ihnen, die Vorgabewerte zu überschreiben, die eingestellt wurden, als Sie sich mit dem Treiber verbunden haben. Ihre Anwendung wird jedoch nicht portierbar sein, wenn Sie diese Optionen verwenden, da sie nicht zur ANSI-Syntax gehören.

Die FROM-Klausel hat folgendes Format:

FROM Tabellenname [Optionen] [Tabellen-Alias]

Die Optionen steuern, welche File-Open-Optionen verwendet werden, ob gelöschte Datensätze zurückgeholt werden und welche Indexdateien geöffnet und für die Sortierung verwendet werden. Die Spezifikation der Optionen lautet:

([COMPATIBILITY= {ANSI | DBASE},]

[CHARSET= {ANSI | IBMPC},]

[RECORDS= {DELETED | UNDELETED},]
[LOCKING= {NONE | RECORD | FILE},]

[Indexangabe])

COMPATIBILITY Nur zur Kompatibilität zu früheren Q+E-Software-Produkten vorhanden.

CHARSET Informiert den Treiber, ob die Daten im ANSI- oder IBM-PC-Zeichensatz gespeichert werden. Sie können im Setup-Dialogfeld den vorgegebenen Zeichensatz einstellen.

RECORDS Bestimmt, ob Datensätze mit oder ohne Löschmarkierung zurückgeliefert werden. Voreingestellt werden Datensätze ohne Löschmarkierung zurückgeliefert.

LOCKING Bestimmt die Ebene der Satzsperrung für die Datenbankdatei. Sie können im Setup-Dialogfeld einstellen, welche Ebene der Satzsperrung voreingestellt ist.

Indexangabe steuert die Verwendung von Indexdateien.

Für .NTX-, .NDX- und .IDX-Dateien lautet das Format:

Indexdatei[/USE], ...

Für .CDX- und .MDX-Dateien ist Indexangabe:

[Indexdatei] [/tag], ...

Sie geben die Liste der Indexdateien an, die geöffnet werden sollen. Der Treiber öffnet automatisch die Produktions- oder strukturierte Indexdatei, wenn diese vorhanden ist.

Tag ist der Name des Indexes innerhalb der Indexdatei, der zur Sortierung der Datensätze verwendet werden soll.

Im folgenden Beispiel wird angegeben, daß die Tabelle "Pers" im IBM-PC-Zeichensatz gespeichert werden soll:

SELECT * FROM Pers (CHARSET=IBMPC, /PersEin)

Indexdateien

Ein *Index* ist eine Datenbankstruktur, die Sie zur Verbesserung der Antwortzeiten von Datenbankabfragen verwenden können. Zu einer Datenbankdatei können ein oder auch mehrere Indizes gehören. Zur Optimierung von Abfragen und zur Pflege der Indizes muß der Treiber alle mit einer Tabelle verbundenen Indizes kennen. Diese Verbindung erfolgt für Produktionsindizes (.CDX und .MDX) und für jeden Index, der in der Datei QEDBF.INI aufgeführt ist. Indizes, die keine Produktionsindizes (.NDX, .NTX und .IDX) und nicht in der Datei QEDBF.INI angegeben sind, müssen explizit in der FROM-Klausel genannt werden, um erkannt zu werden. Der Treiber versucht, unter den verfügbaren Indizes den Treiber zu finden, der sich am besten verwenden läßt. Sie können den Treiber zwingen, einen bestimmten Index zu verwenden, indem Sie den Schalter /USE angeben.

.CDX- und .MDX-Dateien

.CDX- und .MDX-Dateien können mehrere Indizes enthalten. Jeder Index besitzt einen Tagnamen, an dem er erkannt wird. Voreingestellt hat die Indexdatei denselben Namen wie die Datenbankdatei mit der Namenserweiterung .MDX oder .CDX. Falls Sie wissen, welcher Index sich besser verwenden läßt, geben Sie den Tagnamen nach dem Dateinamen der Datenbank in der SELECT-Anweisung an. Ein Beispiel für die Verwendung von dBASE-IV-Indizes:

```
SELECT * FROM Pers (/Pers_nr) WHERE ...
```

Geben Sie den Tagnamen, aber keine ORDER-BY-Klausel an, liefert der Treiber die Datensätze geordnet nach dem Index zurück. Im obigen Beispiel liefert er also die Mitarbeiter geordnet nach ihren Personalnummern.

Der Treiber öffnet automatisch die dBASE-IV-Indexdatei mit dem gleichen Name wie die Datenbankdatei. Geben Sie in der SQL-Anweisung den Namen des Inexes an, der verwendet werden soll, wenn sich der Name der Indexdatei von dem der Datenbankdatei unterscheidet und Sie die Indizes nicht mit QEDBF.INI pflegen. Ein Beispiel:

```
SELECT * FROM Pers (PERS2.MDX)
```

.NTX-, .NDX- und .IDX-Dateien

Jede .NTX-, .NDX- und .IDX-Indexdatei enthält nur einen Index.

Pflegen Sie die Indizes nicht mit der Datei QEDBF.INI, können Sie in der SQL-Anweisung den Namen des Indexes angeben, der verwendet werden soll. Sie können für eine Datenbankdatei mehrere Indexdateien öffnen. Ein Beispiel:

```
SELECT * FROM Pers (PERSEIN.NDX, PERSABT.NDX /USE)
```

Indexdateien pflegen

Indexdateien lassen sich auf zwei verschiedene Arten pflegen: automatisch über die Datei QEDBF.INI oder manuell durch bedarfsweise Angabe in den INSERT- und UPDATE-Anweisungen. Die Pflege der Indexdateien über QEDBF.INI ist effizienter, da der Treiber die Indizes automatisch für Sie aktualisiert und dadurch sicherstellt, daß sie mit den Datensätzen in der Datenbankdatei übereinstimmen.

Alle Produktions- und Strukturindizes werden automatisch gepflegt.

Indizes automatisch pflegen

Wenn Sie die Indizes automatisch pflegen lassen, verwendet der dBASE-Treiber die QEDBF.INI-Datei zur Pflege der Informationen über die dBASE-Dateien in diesem Verzeichnis. Der Treiber pflegt für eine dBASE-Datei die folgenden Informationen:

- Zeichensatz (Char set = CS), dadurch ist der Treiber darüber informatiert, ob die Daten im IBM-PC- oder dem ANSI-Zeichensatz gespeichert sind.
- Indexdateien, die bei einer Änderung der dBASE-Datei zu aktualisieren sind
- Eindeutige ANSI-Indizes, die für die Indexdatei zu pflegen sind

Zur automatischen Pflege der Indizes müssen Sie die Indexattribute definieren, wenn Sie den Treiber einrichten. Das Programm fordert Sie zur Eingabe der Namen der Indexdateien auf, die Sie automatisch pflegen lassen möchten, und erstellt die Datei QEDBF.INI. Wenn Sie eine CREATE- oder DROP-INDEX-Anweisung ausführen, wird QEDBF.INI aktualisiert. Sie müssen bei der Ausführung einer INSERT- oder UPDATE-Anweisung dann keinen Indexnamen angeben.

Indizes manuell pflegen

Sie müssen jedoch einen Indexnamen in den INSERT- oder UPDATE-Anweisungen angeben, wenn Sie die Indizes manuell pflegen. Die Indexdateien werden ansonsten nicht mit den Datensätzen in der Datenbankdatei übereinstimmen.

Format der INSERT-Anweisung

INSERT INTO *Dateiname* [(*Indexdatei* ,...)] [(*Spaltenname*,...)] VALUES (*Ausdr*,...)

Format der UPDATE-Anweisung

UPDATE *Dateiname* [(*Indexdatei*,...)] SET *Spaltenname* = (*Ausdruck*,...) [WHERE *Bedingungen*]

- Führen Sie bei .NTX-, .NDX- und .IDX-Dateien jede Indexdatei für *Dateiname* auf.
- Geben Sie die Indexdateinamen von .CDX- und .MDX-Dateien an, deren Name sich (abgesehen von der Namenserweiterung) von dem Namen der Datenbankdatei unterscheidet. Indexdateien mit übereinstimmendem Namen müssen Sie nicht angeben, da diese vom Treiber automatisch geöffnet werden.

Indexattribute definieren

Da es dBASE zuläßt, daß Sie Indexdateien mit anderen Namen als denen der zugehörigen Datendateien erstellen, sieht der Treiber die Definition der Indizes vor, die gepflegt werden sollen, wenn eine Datendatei geändert wird, und der Indizes, die als eindeutig behandelt werden sollen. Wenn Sie die Indizes automatisch pflegen lassen, aktualisiert das System die Indizes für Sie. Dadurch ist sichergestellt, daß die Indizes mit den Datensätzen in der Datenbankdatei übereinstimmen. Das System stellt auch zur Optimierung von Abfragen Indizes zur Verfügung. Es ist nicht erforderlich, Produktions-.MDX-Dateien oder Struktur-.CDX-Dateien zur Pflege zu markieren, da der Treiber diese Dateien für Sie pflegt. Vielleicht möchten Sie jedoch dieses Verfahren anwenden, um einen Tag als eindeutig zu markieren.

Indizes automatisch pflegen:

1. Klicken Sie im Dialogfeld ODBC dBASE Driver Setup auf Define.

   Es erscheint das Dialogfeld Define File.

2. Wählen Sie die .DBF-Datei aus, und klicken Sie auf OK, um die speziellen Indizes zu definieren.

   Es erscheint das Dialogfeld Define Table.

3. Der obere Bereich des Dialogfelds zeigt den Datenbanknamen und das Verzeichnis an, das die Datendatei enthält. Ist diese Datei im IBM-PC-Zeichensatz abgespeichert, wählen Sie OEM to ANSI Translation.

4. Der untere Bereich des Dialogfelds zeigt die Indexinformation für die Datendatei an. Das Listenfeld Index File erlaubt Ihnen die Auswahl jeder beliebigen Indexdatei im Datenbankverzeichnis. Befindet sich die Indexdatei in einem anderen Verzeichnis, müssen Sie den vollständigen Pfadnamen angeben.

   Zur Festlegung, daß eine Indexdatei eindeutig ist, wählen Sie den Indexnamen aus dem Listenfeld Index File und dann rechts von dem Namen Unique. Entsprechend wählen Sie Maintain, wenn der Index gepflegt werden soll. Das Thema Indexdateien pflegen erläutert die Indexdateipflege.

5. Einen ausgewählten Index mit der Namenserweiterung MDX oder CDX können Sie nicht als eindeutig markieren. Stattdessen können Sie die Tags innerhalb des Index als eindeutig markieren. Dazu wählen Sie den Tagnamen aus dem Listenfeld Tag und dann rechts von diesem Namen Unique.

6. Klicken Sie zur Speicherung dieser Information auf OK, oder klicken Sie zum Abbruch des Dialogfeldes auf Cancel.

CREATE-INDEX-Anweisung

Der von Ihnen erstellte Indextyp wird bestimmt durch den Wert des Attributs CreateType, das Sie im Setup-Dialogfeld einstellen. Bei dem Index kann es sich handeln um einen

- dBASE-III-Index (.NDX)
- dBASE-IV-Index (.MDX)
- Clipper-Index (.NTX)
- FoxBASE-Index (.IDX)
- FoxPro-Index (.CDX)

dBASE-IV-Befehl:

USE Datei1

INDEX ON <Ausdruck> [TO <Indexdatei>] / [TAG <Index-Tag>]

SQL-Anweisung:

CREATE [UNIQUE] INDEX *Indexname* ON *Basis-Tabellenname* (*Feldname* [ASC | DESC] [,*Feldname* [ASC | DESC]] ...)

UNIQUE bedeutet, daß der Treiber einen eindeutigen ANSI-Index über die Spalte erstellt und die Eindeutigkeit der Schlüssel sicherstellt. Die Performance wird durch die Verwendung von eindeutigen Indizes verbessert. Eindeutige ANSI-Indizes unterscheiden sich von eindeutigen dBASE-Indizes. Bei eindeutigen ANSI-Indizes erhalten Sie eine Fehlermeldung, wenn Sie versuchen, einen doppelten Wert in ein indiziertes Feld einzufügen. Dagegen sehen Sie bei eindeutigen dBASE-Indizes keine Fehlermeldung, wenn Sie einen doppelten Wert in ein indiziertes Feld einfügen. Dies liegt daran, daß nur ein Schlüssel in die Indexdatei eingefügt wird.

Indexname ist der Name der Indexdatei. Für FoxPro 2.5 und dBASE IV ist dies ein Tag, der zur Erkennung der Indizes in einer Indexdatei erforderlich ist. Jeder Tabellenindex muß einen eindeutigen Namen haben.

Basis-Tabellenname ist der Name der Datenbankdatei, deren Index zu erstellen ist. Die Namenserweiterung .DBF ist nicht erforderlich, da der Treiber diese automatisch hinzufügt, wenn sie nicht vorhanden ist. Voreingestellt werden die dBASE-IV-Indexdateien mit *Basis-Tabellenname*.MDX und die FoxPro-2.5-Indizes mit *Basis-Tabellenname*.CDX benannt.

Feldname ist der Name einer Spalte in der dBASE-Datei.

ASC weist dBASE an, den Index in aufsteigender (ascending) Reihenfolge zu erstellen. DESC weist dBASE an, den Index in absteigender (descending) Reihenfolge zu erstellen. Voreingestellt werden die Indizes in aufsteigender Reihenfolge sortiert.

Sie können die Liste der Feldnamen durch einen zulässigen dBASE-Indexausdruck ersetzen.

DROP-INDEX-Anweisung

Die Anweisung DROP INDEX löscht einen Index aus der aktuellen Datenbank.

dBASE-IV-Befehl:

ERASE [<Indexdatei>] oder DELETE TAG <Tagname>

SQL-Anweisung:

DROP INDEX {Tabellenname.Indexname}

Tabellenname ist der Name der .DBF-Datei ohne die Namenserweiterung.

For FoxPro 2.5 und dBASE IV: *Indexname* ist der Tag. Ansonsten ist *Indexname* der Name der Indexdatei ohne die Namenserweiterung.

Zum Löschen des Index Persein.ndx geben Sie folgende Anweisung ein:

DROP INDEX Pers.Persein

Pack-Anweisung

Datensätze, die bei einer dBASE-Datei gelöscht werden, werden nicht aus der Datei entfernt. Stattdessen werden sie mit einer Löschmarkierung versehen. Auch durch die Aktualisierung von Memofeldern kann Platz in den Dateien verschwendet werden. Zum Entfernen der gelöschten Datensätze und zur Freisetzung des ungenutzten Platzes aus aktualisierten Memofeldern müssen Sie die Datenbank packen.

Zur Wiederverwendung von Platz in einer dBASE-Datei verwenden Sie die Anweisung Pack. Sie hat folgendes Format:

```
PACK Dateiname
```

Dateiname ist der Name der dBASE-Datei, die gepackt werden soll. Die Namenserweiterung .DBF ist nicht erforderlich; der Treiber fügt die Namenserweiterung automatisch hinzu, wenn sie nicht vorhanden ist. Ein Beispiel:

```
PACK Pers
```

Sie können keine Datei packen, die durch einen anderen Benutzer geöffnet wurde. Außerdem können Sie die Anweisung Pack nicht im manuellen Commit-Modus verwenden.

Die Anweisung Pack macht bei der angegebenen Datei folgendes:

- Alle gelöschten Datensätze werden aus der Datei entfernt.
- Die Verweise auf alle gelöschten Datensätze werden aus den .CDX- und .MDX-Indexdateien gelöscht, die denselben Namen wie die Datendatei haben.
- Die Verweise auf alle gelöschten Datensätze werden aus den .NTX-, .NDX- und .IDX-Dateien gelöscht, die in QEDBF.INI angegeben wurden.
- Unbenutzter Platz in der Memodatei (.DBT oder .FPT) wird gelöscht.

Dateisperre

Sie können mit dem dBASE-Treiber Anwendungen aufbauen und ablaufen lassen, die dBASE-Dateien in einem Netzwerk gemeinsam nutzen. Immer wenn mehrere Benutzer eine Anwendung ablaufen lassen, die auf eine gemeinsam genutzte Datenbankdatei zugreift, ist es wichtig, daß die Anwendungen die Datensätze sperren, die verändert werden. Das Sperren eines Datensatzes hindert andere Benutzer daran, den Datensatz zu sperren, zu aktualisieren oder zu löschen.

Ebenen der Sperre

Der dBASE-Treiber unterstützt drei Ebenen der Datenbanksperre: NONE, RECORD und FILE. Sie können diese Ebenen einstellen

- im Setup-Dialogfeld
- auf der Anweisungsebene in der FROM-Klausel. Ein Beispiel:

  SELECT * FROM Pers (locking=none)

  In diesem Fall wird Ihre Anwendung jedoch nicht portierbar sein.

Keine Sperre (NONE) bietet die beste Performance, ist jedoch nur für Einzelplatz-Umgebungen gedacht.

Das System sperrt die Datenbankdateien bei Datensatz- oder Dateisperren während INSERT-, UPDATE-, DELETE- oder SELECT...FOR-UPDATE-Anweisungen. Die Sperren werden freigegeben, wenn der Benutzer die Transaktion abschließt. Die Sperren hindern andere Benutzer daran, die gesperrten Objekte zu ändern, aber sie sperren keine Leser aus.

Bei der Satzsperre werden nur Datensätze gesperrt, die von der Anweisung betroffen sind. Die Satzsperre erlaubt einen besseren gleichzeitigen Betrieb mit anderen Benutzern, die ebenfalls die Tabelle ändern möchten.

Die Dateisperre sperrt die gesamte Datei. Der Verwaltungsaufwand bei der Dateisperre ist niedriger und die Dateisperre kann besser funktionieren, wenn Datensätze nicht häufig geändert werden, nur ein Benutzer die Datensätze ändert oder eine große Anzahl Datensätze geändert wird.

Sperren auf lokale Dateien anwenden

Wenn Sie die Datenbanksperre verwenden und auf lokale Dateien (nicht Dateien auf einem Netzwerk-Server) zugreifen, sollten Sie das DOS-Dienstprogramm SHARE.EXE ablaufen lassen, bevor Sie Windows aufrufen. Das Programm wird automatisch bei jedem Start Ihres Computers ausgeführt, wenn Sie SHARE.EXE zu Ihrer AUTOEXEC.BAT-Datei hinzufügen.

Grenze bei der Anzahl der Sperren

Es gibt eine Grenze für die Anzahl der Sperren, die auf eine Datei angewandt werden können. Falls Sie auf eine dBASE-Datei auf einem Server zugreifen, ist die Anzahl der Sperren vom Server abhängig (sehen Sie in der Dokumentation zu Ihrem Server nach). Greifen Sie auf eine lokale dBASE-Datei zu, hängt diese Grenze vom zugewiesenen Pufferplatz ab, der beim Laden von SHARE.EXE zugewiesen wurde (mehr Informationen dazu finden Sie in Ihrer DOS-Dokumentation). Überschreiten Sie die Anzahl der verfügbaren Sperren, können Sie zur Dateisperre wechseln.

Kompatibilität bei den Sperren

Der dBASE-Treiber unterstützt mehrere unterschiedliche Sperrschemen.

Alle Anwendungen, die Tabellen gemeinsam mit anderen nutzen, müssen dasselbe Sperrschema verwenden. Wählen Sie zum Beispiel die Clipper-Sperre, wenn Sie beabsichtigen, Ihre Tabellen mit Clipper-Anwendungen gemeinsam zu nutzen.

Q+EVIRTUAL arbeitet genauso wie die ursprüngliche Sperre von Q+E; die Dateien werden jedoch nicht mehr länger exklusiv gesperrt. Bei der ursprünglichen Q+E-Sperre konnten zwar andere Q+E-Anwendungen die Datenbank lesen und in diese schreiben, aber alle Anwendungen von Fremdanbietern

wurden von der Datei vollständig ausgesperrt (selbst vom Lesen anderer Datensätze).

Stellen Sie die DBASE-, CLIPPER- und FOX-Sperren ein, wenn Sie eine Q+E-Anwendung mit einer dBASE-, Clipper- oder FoxPro-Anwendung verwenden und möchten, daß die Sperrmechanismen kompatibel sind.

Die Verwendung eines Q-E-Sperrschemas hat gegenüber dem dBASE-Pendant den Vorteil, daß Q+E in einem Index nur individuelle Tags sperrt, wohingegen es für dBASE erforderlich ist, den gesamten Index bei Einfügungen und Aktualisierungen zu sperren.

Wie sich Transaktionen auf Satzsperren auswirken

Wenn eine UPDATE- oder DELETE-Anweisung ausgeführt wird, sperrt der Treiber die Datensätze, die von dieser Anweisung betroffen sind. Die Sperren werden freigegeben, nachdem der Treiber der Datenbank die Änderungen abgeschlossen hat. Im manuellen Commit-Modus bedeutet dies, daß die Sperren aufrechterhalten werden, bis die Anwendung die Transaktion abschließt. Im Autocommit-Modus bedeutet dies, daß die Sperren aufrechterhalten werden, bis die Anweisung ausgeführt ist.

Bei einer SELECT...FOR-UPDATE-Anweisung sperrt der Treiber den Datensatz nur, wenn dieser geholt wird. Nachdem der Datensatz aktualisiert ist, hält der Treiber die Sperre aufrecht, bis die Änderungen an die Datenbank übergeben sind. Ansonsten wird die Sperre freigegeben, wenn der nächste Datensatz geholt wird.

Unterstützte SQL-Anweisungen

Dieses Hilfethema erörtert die folgenden SQL-Anweisungen in der Form, wie sie durch den dBASE-Datenbanktreiber unterstützt werden.

SELECT

CREATE TABLE

CREATE INDEX

DROP INDEX

DROP TABLE

INSERT

UPDATE

DELETE

SELECT

Die Anweisung SELECT wählt Zeilen aus Tabellen aus, um diese entweder anzuzeigen oder sie als Eingabe für andere SQL-Anweisungen zu verwenden.

dBASE-IV-Befehl:

```
USE Pers

DISPLAY ALL Nachname, Vorname FOR Gehalt > 1000
```

SQL-Anweisung:

```
SELECT [DISTINCT] {* | Spaltenausdruck, ...}

    FROM Dateiangabe, ...

    [ WHERE Bedingungen ]

    [ GROUP BY {Spaltenausdruck, ...} ]
    [ HAVING {Bedingungen} ]

    [ UNION [ALL] (SELECT...) ]

    [ ORDER BY {Sortierausdruck  [DESC | ASC]}, ... ]

    [ FOR UPDATE OF {Spaltenausdruck, ...}]
```

Geben Sie hinter dem Schlüsselwort SELECT eine Liste von *Spaltenausdrücken* an, die Sie abfragen möchten, oder ein Sternchen (*) zur Abfrage aller Felder.

Der gebräuchlichste *Ausdruck* ist einfach ein Feldname (zum Beispiel: NACHNAME). Kompliziertere Ausdrücke können mathematische Operationen der Zeichenfolgen-Manipulationen beinhalten (zum Beispiel GEHALT * 1.05).

Mehrere *Spaltenausdrücke* trennen Sie durch Kommas (zum Beispiel: NACHNAME, VORNAME, EINSTDATUM).

Vor Feldnamen kann der Tabellenname oder ein Aliasname angegeben werden. Zum Beispiel: PERS.NACHNAME oder P.NACHNAME, wobei P der ALiasname für die Tabelle PERS ist.

DISTINCT-Operator

Der Operator DISTINCT kann vor dem ersten Spaltenausdruck stehen. Dieser Operator entfernt doppelte Zeilen aus dem Ergebnis einer Abfrage. Ein Beispiel:

```
SELECT DISTINCT Nachname FROM Pers
```

FROM-Klausel

Geben Sie hinter FROM eine Liste von Dateispezifikationen an (Dateispez). Dateispezifikationen haben das Format:

FROM *Tabellenname* [*Optionen*] [*Tabellenalias*]

Tabellenname kann ein einfacher Tabellenname im aktuellen Arbeitsverzeichnis oder ein vollständiger Pfadname (C:\ODBC\PERS) sein.

Optionen sind von der Datenbank abhängig. Zum Beispiel ist für dBASE eine Option die Ebene der verwendeten Satzsperre.

Tabellenalias ist ein Name, der im Rest der SELECT-Anweisung zum Bezug auf diese Tabelle verwendet wird. Der Tabellenalias kann vor den Feldnamen der Datenbank angegeben werden. Bei der Tabellenspezifikation

FROM Pers P

können Sie auf das Feld NACHNAME als P.NACHNAME Bezug nehmen. Tabellenaliasnamen müssen verwendet werden, wenn die SELECT-Anweisung eine Tabelle mit sich selbst verbindet. Ein Beispiel:

```
SELECT * FROM Pers P, Pers F
WHERE P.Mgr_Nr = F.Pers_Nr
```

WHERE-Klausel

Die WHERE-Klausel legt die Bedingungen fest, die Datensätze erfüllen müssen, um in die Abfrage aufgenommen zu werden.

dBASE-IV-Befehl:

... FOR <Ausdruck>

SQL-Anweisung:

WHERE *Ausdr1 rel_Operator Ausdr2*

Ausdr1 und *Ausdr2* können Feldnamen, konstante Werte oder Ausdrücke sein. *Rel_Operator* ist der relationale Operator, der die zwei Ausdrücke miteinander verbindet.

Zum Beispiel werden mit der folgenden SELECT-Anweisung die Namen der Beschäftigten abgefragt, die ein Jahreseinkommen von mindestens 34.000 DM haben:

```
SELECT Nachname,Vorname FROM Pers
WHERE Gehalt >= 34000
```

GROUP-BY-Klausel

Die GROUP-BY-Klausel gibt den Namen des oder der Felder an, nach denen die Rückgabewerte gruppiert werden sollen. Diese Klausel wird verwendet, um einen Satz von Aggregatwerten zurückzuerhalten.

dBASE-IV-Befehl:

REPORT ... SUMMARY

SQL-Anweisung:

GROUP BY *Spaltenausdrücke*

Spaltenausdrücke können ein oder mehrere Feldnamen der Datenbanktabelle sein, die durch ein Komma (,) voneinander getrennt sind, oder ein oder mehrere Ausdrücke, die durch ein Komma (,) voneinander getrennt sind.

Das folgende Beispiel summiert die Gehälter in den einzelnen Abteilungen auf.

```
SELECT Abtlgs_Nr, sum(Gehalt)
FROM Pers
GROUP BY Abtlgs_Nr
```

Diese Anweisung liefert eine Zeile für jede unterschiedliche Abteilungsnummer. Jede Zeile enthält die Abteilungsnummer und die Summe der Gehälter der Mitarbeiter in der Abteilung.

HAVING-Klausel

Die HAVING-Klausel ermöglicht es Ihnen, Bedingungen für Datensatzgruppen anzugeben (zum Beispiel: "Zeige nur die Abteilungen mit Gehältern, deren Gesamtsumme 340.000 DM überschreitet"). Diese Klausel ist nur dann zulässig, wenn Sie bereits eine GROUP-BY-Klausel spezifiziert haben.

dBASE-IV-Befehl:

REPORT... SUMMARY

SQL-Anweisung:

HAVING *Ausdr1 rel_Operator Ausdr2*

Ausdr1 und *Ausdr2* können Feldnamen, konstante Werte oder Ausdrücke sein. *Rel_Operator* ist der relationale Operator, der die zwei Ausdrücke verbindet.

Das folgende Beispiel liefert nur die Abteilungen zurück, deren Summe der Gehälter 340.000 DM übersteigt.

```
SELECT Abtlgs_Nr, sum(Gehalt)
FROM Pers
GROUP BY Abtlgs_Nr
HAVING sum(Gehalt) > 340000
```

UNION-Operator

Der UNION-Operator kombiniert die Ergebnisse von zwei SELECT-Anweisungen in einem einzigen Ergebnis. Das eine Ergebnis besteht aus allen zurückgelieferten Datensätzen beider SELECT-Anweisungen. Voreingestellt werden doppelte Datensätze nicht zurückgeliefert. Zur Rückgabe doppelter Datensätze verwenden Sie das Schlüsselwort ALL (UNION ALL). Das Format ist:

SELECT *Anweisung*
UNION [ALL]
SELECT *Anweisung*

Wenn Sie den UNION-Operator verwenden, müssen die Auswahllisten für die SELECT-Anweisungen die gleiche Anzahl von Spaltenausdrücken mit den gleichen Datentypen und in der gleichen Reihenfolge haben. Ein Beispiel:

```
SELECT Nachname, Gehalt, EinstDatum
FROM Pers
UNION
SELECT Name, Lohn, Gebdatum
FROM Person
```

Dieses Beispiel hat die gleiche Anzahl von Spaltenausdrücken, und jeder Spaltenausdruck hat in der Reihenfolge denselben Datentyp.

Das folgende Beispiel ist *nicht* zulässig, da sich die Datentypen der Spaltenausdrücke unterscheiden (GEHALT aus PERS hat einen anderen Datentyp als NACHNAME von ERHÖHUNG). Dieses Beispiel hat die gleiche Anzahl an Spaltenausdrücken in jeder SELECT-Anweisung, aber die Ausdrücke sind nach Datentypen nicht in der gleichen Reihenfolge geordnet.

```
SELECT Nachname, Gehalt
FROM Pers
UNION
SELECT Gehalt, Nachname
FROM Erhöhung
```

ORDER-BY-Klausel

Die Klausel ORDER BY zeigt, wie die Datensätze zu sortieren sind.

dBASE-IV-Befehl:

```
USE Pers INDEX Persnr

DISPLAY Persnr,Nachname
```

SQL-Anweisung:

ORDER BY {*Sortierausdruck* [DESC | ASC]}, ...

Sortierausdruck können Feldnamen, Ausdrücke oder die Positionsnummer des zu verwendenden Spaltenausdrucks sein. Zum Beispiel können Sie zum Sortieren nach NACHNAME eine der folgenden SELECT-Anweisungen verwenden:

```
SELECT Persnr, Nachname, Vorname FROM Pers
ORDER BY Nachname
```

oder

```
SELECT Persnr, Nachname, Vorname FROM Pers
ORDER BY 2
```

Im zweiten Beispiel ist NACHNAME der zweite Spaltenausdruck, der SELECT folgt, so daß ORDER BY 2 nach NACHNAME sortiert.

FOR-UPDATE-OF-Klausel

Die Klausel FOR UPDATE OF sperrt die Datenbankdatei, die durch die SELECT-Anweisung ausgewählt wurden. Das Format ist:

FOR UPDATE OF *Spaltenausdrücke*

Spaltenausdrücke ist eine Liste von Feldnamen in der Datenbankdatei, die Sie zu aktualisieren beabsichtigen und die durch ein Komma (,) voneinander abgetrennt sind.

Das folgende Beispiel liefert alle Datensätze der Personal-Datenbank zurück, die einen Wert von über 34.000 DM im Feld GEHALT stehen haben. Jeder Datensatz, der geholt wird, wird gesperrt. Die Sperre wird freigegeben, wenn Sie ein positioniertes Update vornehmen oder mit einer "aktuellen Cursorposition"-Anweisung löschen und die Änderung oder das Löschen mit COMMIT übernehmen. Falls der Datensatz nicht aktualisiert oder gelöscht ist, wird die Sperre freigegeben, wenn Sie den nächsten Datensatz holen.

```
SELECT *
FROM Pers
WHERE Gehalt > 34000
FOR UPDATE OF Nachname, Vorname, Gehalt
```

CREATE-TABLE-Anweisung

dBASE-IV-Befehl:

CREATE Tabellenname

SQL-Anweisung:

CREATE TABLE *Dateiname* (*Spaltendefinition*, *Spaltendefinition* ...)

Der *Dateiname* kann ein einfacher Dateiname (PERS) oder ein vollständiger Pfadname sein (C:\ODBC\PERS).

Spaltendefinition ist der Spaltenname, gefolgt von dem Datentyp. Die zulässigen Werte für Spaltennamen hängen von der Datenbank ab.

Datentyp ist die Angabe des Datentyps einer Spalte.

> **Hinweis:** In den Definitionen von NUMERIC- und FLOAT-Feldern gibt die erste Zahl die Anzahl der ganzzahligen Ziffern und die zweite Zahl die Anzahl der Dezimalstellen an. Dies ist keine Definition der Gesamtlänge mit Dezimalstellen wie bei dBASE. Falls keine Dezimalstellen vorhanden sind, kann für NUMERIC- und FLOAT-Felder eine Länge von 20 angegeben werden. Sind Dezimalstellen vorhanden, darf die kombinierte Anzahl von Ganzzahl- und Dezimalstellen 19 nicht überschreiten.

Ein Beispiel für eine CREATE-TABLE-Anweisung zum Erstellen einer Personal-Datenbankdatei:

CREATE TABLE Pers (Nachname CHAR(20), Vorname CHAR(12), Gehalt NUMERIC (10,2), Einstdatum DATE)

DROP-TABLE-Anweisung

Die Anweisung DROP TABLE löscht eine Tabelle aus der aktuellen Datenbank.

dBASE-IV-Befehl:

    ERASE Tabellenname

SQL-Anweisung:

    DROP TABLE *Dateiname*

Der *Dateiname* kann ein einfacher Dateiname (PERS) oder ein vollständiger Pfadname (C:\ODBC\PERS) sein.

Ein Beispiel für eine DROP-TABLE-Anweisung zum Löschen der Personal-Datenbank:

    DROP TABLE Pers

INSERT-Anweisung

Die SQL-Anweisung INSERT wird verwendet, um neue Datensätze zu einer Datenbankdatei hinzuzufügen.

dBASE-IV-Befehl:

```
USE Datei1
APPEND
```

SQL-Anweisung:

```
INSERT INTO Dateiname [Optionen] [(Spaltenname,...)] VALUES (Ausdr, ...)
```

Die Listen der Indizes und der Spaltennamen sind optional.

Der *Dateiname* kann ein einfacher Dateiname (PERS) oder ein vollständiger Pfadname (C:\ODBC\ PERS) sein.

Die *Optionen* sind von der Datenbank abhängig.

Die *Spaltenname*-Liste ist eine optionale Liste von Spaltennamen mit den Namen und der Reihenfolge der Spalten, deren Werte in der Values-Klausel angegeben sind. Wenn Sie Spaltenname weglassen, müssen die Ausdrücke (*Ausdr*) Werte für alle in der Datei definierten Spalten enthalten und in derselben Reihenfolge sein wie die Spalten, die für die Datei definiert sind.

Die *Ausdr*-Liste ist die Liste von Ausdrücken, die die Werte für die Spalten des neuen Datensatzes liefert. Normalerweise sind die Ausdrücke konstante Werte für die Spalten. Zeichenfolgen-Werte müssen in einfache oder doppelte Anführungszeichen, Datumswerte in geschweifte Klammern {} und logische Werte in Punkte (zum Beispiel .T. oder .F.) eingeschlossen sein.

Ein Beispiel einer INSERT-Anweisung auf die Personal-Datei:

```
INSERT INTO Pers (Nachname, Vorname, Persnr, Gehalt, Einstdatum)
VALUES ('Schmidt', 'Johannes', 'E22345', 27500, {4.6.91})
```

Jede INSERT-Anweisung fügt einen Datensatz zu der Datenbankdatei hinzu. In diesem Fall wurde ein Datensatz zu der Personal-Datenbankdatei PERS hinzugefügt. Es werden Werte für fünf Spalten angegeben. Den restlichen Spalten in der Datei wird ein Leerwert zugewiesen, also Null. Beachten Sie, daß Zeichenwerte in Anführungsstriche und Daten in geschweifte Klammern ({}) eingeschlossen werden müssen.

UPDATE-Anweisung

Die SQL-Anweisung UPDATE wird zum Ändern von Datensätzen in einer Datenbankdatei verwendet.

dBASE-IV-Befehl:

USE Datei1

REPLACE <Feld> with <Ausdruck> [,...] FOR <Bedingung>

SQL-Anweisung:

UPDATE *Dateiname* [*Optionen*] SET *Spaltenname* = [*Ausdr*, ...]
[WHERE { *Bedingungen* | CURRENT OF *Cursorname* }]

Dateiname kann ein einfacher Dateiname (PERS) oder ein vollständiger Pfadname (C:\ODBC\PERS) der Datei sein, die zu aktualisieren ist.

Die *Optionen* sind von der Datenbank abhängig.

Spaltenname ist der Name einer Spalte, deren Wert zu ändern ist. Mehrere Spalten können in einer Anweisung geändert werden.

Ausdr ist der neue Wert für die Spalte. Der Ausdruck kann ein konstanter Wert oder eine Unterabfrage sein. Zeichenfolgen-Werte müssen in einfache oder doppelte Anführungszeichen, Datumswerte in geschweiften Klammern ({}) und logische Werte in Punkte (zum Beispiel .T. oder .F.) eingeschlossen werden. Unterabfragen müssen in Klammern eingeschlossen werden.

Die WHERE-Klausel bestimmt, welche Datensätze zu aktualisieren sind.

Die Klausel WHERE CURRENT OF *Cursorname* kann nur von Entwicklern verwendet werden, die direkt die ODBC API programmieren. Sie bewirkt, daß die Zeile, an der *Cursorname* positioniert ist, aktualisiert wird. Dies wird als positioniertes Update" bezeichnet. Sie müssen zuerst eine SELECT...FOR-UPDATE-Anweisung mit einem benannten Cursor ausführen und die Zeile holen, die aktualisiert werden soll.

Ein Beispiel einer UPDATE-Anweisung auf die Personal-Datei:

UPDATE Pers SET Gehalt=32000, Experst=.T. WHERE Persnr = 'E10001'

Die UPDATE-Anweisung ändert jeden Datensatz, der die Bedingungen in der WHERE-Klausel erfüllt. In diesem Fall wird das Gehalt und der Experst-Status für alle Mitarbeiter geändert, deren Personalnummer E10001 ist. Da die Personalnummern in der Personaldatei eindeutig sind, wird ein Datensatz aktualisiert.

Ein ,eispiel für die Anwendung einer Unterabfrage:

UPDATE Pers SET Gehalt = (select avg(Gehalt) from Pers) where Persnr = 'E10001'

In diesem Fall wird das Gehalt für den Mitarbeiter mit der Personalnummer E10001 in das Durchschnittsgehalt in der Firma geändert.

DELETE-Anweisung

Die SQL-Anweisung DELETE wird verwendet, um Datensätze aus einer Datenbankdatei zu löschen.

dBASE-IV-Befehl:

USE Datei1

DELETE ... FOR <Bedingung>

SQL-Anweisung:

DELETE FROM *Dateiname* [*Optionen*]
[WHERE {*Bedingungen* | CURRENT OF *Cursorname* }]

Dateiname ist der Name der Datenbankdatei, deren Datensätze gelöscht werden sollen. Der Dateiname kann ein einfacher Dateiname (PERS) oder ein vollständiger Pfadname (C:\ODBC\PERS) sein.

Die *Optionen* sind abhängig von der Datenbank.

Die WHERE-Klausel bestimmt, welche Datensätze zu löschen sind.

Die Klausel WHERE CURRENT OF *Cursorname* kann nur von Entwicklern verwendet werden, die direkt die ODBC API programmieren. Sie bewirkt, daß die Zeile, an der *Cursorname* positioniert ist, gelöscht wird. Dies wird als positioniertes Löschen" bezeichnet. Sie müssen zuerst eine SELECT...FOR-UPDATE-Anweisung mit einem benannten Cursor ausführen und die Zeile holen, die gelöscht werden soll.

Ein Beispiel für eine DELETE-Anweisung auf die Personaldatei:

DELETE FROM Pers WHERE Persnr = 'E10001'

Jede DELETE-Anweisung entfernt jeden Datensatz, der die Bedingungen in der WHERE-Klausel erfüllt. In diesem Fall wird jeder Datensatz, der die Personalnummer E10001 hat, gelöscht. Da die Personalnummern in der Personaldatei eindeutig sind, wird ein Datensatz gelöscht.

Performance

Dieser Abschnitt behandelt die Verfahren, mit denen Sie die Performance von Datenbankabfragen verbessern können. Dies umfaßt die folgenden Themen:

Indexdateien

Performance bei der Datensatzauswahl verbessern

Mehrfachfelder indizieren

Performance bei Verbindungen verbessern

Indexdateien

Ein Datenbanktreiber kann Indizes verwenden, um Datensätze schnell zu finden. Zum Beispiel verringert ein Index auf das Feld Persnr die Zeit erheblich, die der Treiber für die Suche nach einer bestimmten Personalnummer benötigt. Betrachten Sie die folgende WHERE-Klausel:

```
WHERE Persnr = 'E10001'
```

Ohne Index muß der Treiber die gesamte Datenbanktabelle durchsuchen, um die Datensätze mit der Personalnummer E10001 zu finden. Durch die Verwendung eines Index auf das Feld Persnr kann der Treiber jedoch unter Verwendung der Indexverweise schnell die Datensätze finden.

Indexdateien können die Performance von SELECT-Anweisungen verbessern. Bei kleinen Dateien werden Sie diese Verbesserung vielleicht gar nicht wahrnehmen, aber sie kann bei großen Dateien bemerkenswert sein. Zu viele Indizes sind jedoch von Nachteil. Indizes verlangsamen die Performance des Einfügens, Aktualisierens und Löschens von Datensätzen, da der Treiber zusätzlich zu den Datenbankdateien noch die Indizes zu verwalten hat. Außerdem benötigen Indizes zusätzlichen Festplattenplatz.

Performance bei der Datensatzauswahl verbessern

Damit Indizes die Performance von Auswahlen verbessern können, muß der Indexausdruck exakt mit der Auswahlbedingung übereinstimmen. Haben Sie zum Beispiel einen Index mit dem Ausdruck NACHNAME erstellt, verwendet die folgende SELECT-Anweisung diesen Index:

```
SELECT * FROM Pers WHERE Nachname = 'Schmidt'
```

Diese SELECT-Anweisung verwendet den Index dagegen nicht:

```
SELECT * FROM Pers WHERE UPPER(Nachname) = 'Schmidt'
```

Die zweite Anweisung verwendet den Index nicht, da die WHERE-Klausel UPPER(Nachname) enthält und dieser Ausdruck nicht mit dem Indexausdruck NACHNAME übereinstimmt. Wenn Sie vorhaben, die UPPER-Funktion in allen Ihren SELECT-Anweisungen zu verwenden, und Ihre Datenbank Indizes auf Ausdrücke unterstützt, dann sollten Sie einen Index mit dem Ausdruck UPPER(Nachname) definieren.

Mehrfachfelder indizieren

Falls Sie häufiger WHERE-Klauseln verwenden, die mehr als ein Feld beinhalten, möchten Sie vielleicht einen Index erstellen, der mehrere Felder enthält. Betrachten Sie die folgende WHERE-Klausel:

```
WHERE Nachname = 'Schmidt' und Vorname =   'Thomas'
```

Für diese Bedingung ist der optimale Indexfeldausdruck NACHNAME,VORNAME. Dieser Ausdruck erstellt einen verketteten Index.

Verkettete Indizes können auch für WHERE-Klauseln verwendet werden, die nur das erste von zwei verketteten Feldern enthalten. Der Index NACHNAME, VORNAME verbessert auch die Performance der folgenden WHERE-Klausel (obwohl kein Wert für Vorname angegeben ist):

```
Nachname = 'Schmidt'
```

Ein Treiber verwendet bei der Verarbeitung von WHERE-Klauseln nur einen Index. Haben Sie komplizierte WHERE-Klauseln mit einer Anzahl Bedingungen für unterschiedliche Felder und Indizes auf mehrere der Felder, wählt der Treiber einen Index davon aus, den er verwendet. Der Treiber wird zuerst Indizes auf Bedingungen verwenden, die das Gleichheitszeichen (=) als Vergleichsoperator haben. Nehmen Sie als Beispiel an, Sie hätten je einen Index auf das Feld Persnr sowie auf das Feld Nachname und die folgende WHERE-Klausel:

```
WHERE Persnr >= 'E10001' AND Nachname = 'Schmidt'
```

Der Treiber wählt den Index auf das Feld Nachname.

Falls keine Bedingung ein Gleichheitszeichen enthält, versucht der Treiber zuerst, einen Index mit einer Bedingung, die eine untere *und* eine obere Grenze hat, zu verwenden und dann einen Index mit einer Bedingung, die eine untere *oder* eine obere Grenze hat. Der Treiber wird immer versuchen, den Index zu finden, der die Auswahl am meisten einschränkt und die WHERE-Klausel erfüllt.

In den meisten Fällen verwendet der Treiber keinen Index, wenn die WHERE-Klausel den Vergleichsoperator OR enthält. Zum Beispiel verwendet der Treiber bei der folgenden WHERE-Klausel keinen Index:

```
WHERE Persnr >= 'E10001' OR Nachname = 'Schmidt'
```

Performance bei Verbindungen verbessern

Wenn Datenbankdateien miteinander verbunden werden, können Indexdateien die Performance erheblich verbessern. Stehen die richtigen Indizes nicht zur Verfügung, können Abfragen, die Verbindungen verwenden, lange Zeit dauern.

Nehmen Sie an, Sie hätten die folgende SELECT-Anweisung:

```
SELECT * FROM Abtlg, Pers WHERE Abtlg.Abtlgnr = Pers.Abtlg
```

In diesem Beispiel sind die Datenbankdateien Abtlg und Pers über das Feld Abtlgnr miteinander verbunden. Führt der Treiber eine Abfrage aus, die eine Verbindung enthält, verarbeitet er die Tabellen von links nach rechts und verwendet einen Index auf das Verbindungsfeld der zweiten Tabelle (das Feld ABTLG der Datei PERS).

Zur Verbesserung der Performance während der Verbindung benötigen Sie einen Index auf das Verbindungsfeld der zweiten Datei in der FROM-Klausel. Gibt es eine dritte Datei in der FROM-Klausel, verwendet der Treiber außerdem einen Index auf das Feld in der dritten Datei, die es mit einer beliebigen vorherigen Datei verbindet. Ein Beispiel:

```
SELECT * FROM Abtlg, Pers, Addr
WHERE Abtlg.Abtlgnr = Pers.Abtlg AND Pers.Ort = Addr.Ort
```

In diesem Fall sollten Sie einen Index auf das Feld BERS.ABTLG und das Feld ADDR.ORT haben.

dBASE-Datentypen

dBASE-IV-Datentypen

dBASE-III- und Clipper-Datentypen

FoxPro- und FoxBASE-Datentypen

ODBC-dBASE-Driver-Setup-Dialogfeld

Hinweis Dieser Treiber ist nur für die Verwendung mit Lotus-Anwendungen lizensiert.

Data Source Name

Kennzeichnet eine einzelne Verbindung zu einer dBASE-Datenbank. Dies kann eine beliebige Zeichenfolge sein. Beispiele: "Buchhaltung" oder "dBASE-Dateien".

Description

Eine optionale lange Beschreibung eines Datenquellennamens. Beispiele: "Meine Konten-Datenbank" oder "dBASE-Dateien in Verzeichnis C:\DBF".

Database Directory

Das Verzeichnis, in dem die dBASE-Dateien abgespeichert sind. Falls kein Verzeichnis angegeben ist, wird das aktuelle Arbeitsverzeichnis verwendet.

Create Type

Gibt an, welches dBASE-Format der Treiber zum Erstellen neuer Tabellen verwenden soll. Wählen Sie dBASE 2, dBASE 3, dBASE 4, Clipper, FoxBase, FoxPro1 oder FoxPro25.

Lock Compatibility

Bestimmt das physikalische Sperren der Datenbankdatei. Wählen Sie DBASE, Q+E, Q+EVIRTUAL, CLIPPER oder FOX.

Locking

Bestimmt die Sperrebene für die Datenbankdateien (NONE, RECORD oder FILE). LCK=NONE bietet die beste Performance, ist jedoch nur für Einzelbenutzerumgebungen gedacht. Mit LCK=RECORD werden nur die Datensätze gesperrt, die von einer Anweisung betroffen sind. LCK=FILE sperrt die gesamte Datei.

File Open Cache

Bestimmt die maximale Anzahl ungenutzter Dateien, die geöffnet bleiben. Öffnet ein Benutzer zum Beispiel bei FOC=4 vier Dateien und schließt diese wieder, sind die Dateien nicht tatsächlich geschlossen. Der Treiber hält die Dateien offen, so daß er kein weiteres zeitraubendes Öffnen der Dateien durchführen muß, wenn eine andere Abfrage eine dieser Dateien verwendet. Der FOC bietet den Vorteil einer verbesserten Performance. Der Nachteil ist, daß ein Benutzer, der versucht, die Datei exklusiv zu öffnen, einen Sperrkonflikt erhält, obwohl scheinbar niemand die Datei geöffnet hat.

Cache Size

Bestimmt die Anzahl der 64K-Blöcke, die der Treiber verwendet, um Datenbank-Datensätze zwischenzuspeichern. Die Performance ist umso höher, je höher die Anzahl der Blöcke ist. Die Voreinstellung ist 4 Blöcke. Es hängt vom verfügbaren Systemspeicher ab, wie viele Blöcke Sie maximal einstellen können. Wenn Sie rückwärtsblättern und CacheSize ist >0, können Sie keine Aktualisierungen anderer Benutzer sehen, bevor Sie die SELECT-Anweisung nicht erneut ausgeführt haben.

International Sort

Bestimmt die Reihenfolge, in der die Datensätze abgefragt werden, wenn Sie eine SELECT-Anweisung erteilen. Wählen Sie International Sort, wenn Sie die internationale Sortierreihenfolge verwenden möchten, so wie sie durch Windows definiert ist. Die Sortierreihenfolge unterscheidet die Groß-/Kleinschreibung (*a* steht vor *B*). Die Sortierung von Zeichen mit Akzent ist ebenso berücksichtigt. (Weitere Informationen finden Sie in Ihrer Windows-Dokumentation.) Wählen Sie International Sort nicht, wenn Sie die ASCII-Sortierreihenfolge verwenden möchten, bei der Großbuchstaben vor

Kleinbuchstaben stehen (*B* steht vor *a*).

Perform OEM to ANSI Translation

Bestimmt, ob die Datenbankdatei den ANSI- oder IBM-PC-Zeichensatz verwendet. Die zwei Zeichensätze sind abgesehen von den internationalen Zeichen gleich. Der ANSI-Zeichensatz unterstützt internationale Zeichen besser als der IBM-PC-Zeichensatz.

Define...:

Wählen Sie Define , um Ihre dBASE-Dateien mit Indexdateien zu verbinden.

Indexattribute definieren

Der obere Abschnitt des Dialogfeldes Define Table zeigt den Namen und das Verzeichnis, in dem sich die dBASE-Datei befindet. Wählen Sie OEM to ANSI Translation, wenn diese Datei den IBM-PC-Zeichensatz verwendet.

Der untere Abschnitt des Dialogfeldes zeigt alle Indexdateien in diesem Verzeichnis an. Wählen Sie jede Indexdatei, die Sie mit der dBASE-Datei verbinden möchten, und wählen Sie Maintain. Falls die Indexdatei eindeutig ist, kreuzen Sie das Kontrollfeld Unique rechts neben dem Dateinamen an.

Hat der ausgewählte Index eine .MDX- oder .CDX-Namenserweiterung, können Sie die Indexdatei nicht als eindeutig markieren. Sie können jedoch die Tags innerhalb des Indexes als eindeutig markieren. Dazu wählen Sie den Tagnamen aus dem Listenfeld Tag und kreuzen das Feld Unique rechts von dem Namen an.