Kurzeinführung ODBC

ODBC (Open Database Connectivity) ist eine von Microsoft entworfene Anwendungsschnittstelle (Application Programming Interface, API), welche in der Lage ist, auf alle gebräuchlichen Datenbankmanagementsysteme (DBMS) zuzugreifen. Der Aufbau und die Funktion von ODBC soll im Folgenden dargestellt werden. Dabei soll zuerst in einem Codebeispiel eine Datenbankoperation in ODBC einer Datenbankoperation in Embedded SQL gegenübergestellt werden.

1 Codebeispiele

1.1 Embedded SQL

Das Programm in Abbildung 1 zeigt ein C-Programm mit eingebettetem SQL. Es werden alle Zeilen der Angestelltentabelle `EMP` (mit Spalten für Name (`EMPNAME`) und Telefonnummer (`TELNO`) der Angestellten) ermittelt, welche mit einer bestimmten Telefonnummer übereinstimmen. Bei diesem Codebeispiel wird auf die Darstellung des Auf- und Abbaus der Verbindung zum DBMS verzichtet.

```
1  /* Deklaration der Hostvariablen. Über sie erfolgt der Austausch */
2  /* der Daten zwischen Datenbankoperation und C-Programm          */
3  EXEC SQL BEGIN DECLARE SECTION;
4  char NAME[26];
5  short TNUM;
6  EXEC SQL END DECLARE SECTION;
7
8  main()
9  {
10 /* Abfrage der Zeilen, die mit einer best. Tel.nummer übereinstimmen  */
11 /* C1 stellt einen Zeiger auf die aus der Abfrage resultierenden      */
12 /* Zeilen dar. Er wird in einer while-Schleife zeilenweise ausgelesen */
13 EXEC SQL DECLARE C1 CURSOR FOR
14    SELECT EMPNAME, TELNO FROM EMP WHERE TELNO > 7301;
15 EXEC SQL OPEN C1
16
17 while (SQLCODE == 0)
18     {
19     EXEC SQL FETCH C1 INTO :NAME, TNUM;
20     }
21 EXEC SQL CLOSE C1
22 }
```

Abbildung 1 Codebeispiel Embedded SQL

Diese Embedded-SQL Anwendung wird nun durch einen Precompiler übersetzt, bevor sie vom Compiler der Host-Programmiersprache (Programmiersprache, in welche die SQL-Anweisungen eingebettet sind) übersetzt werden kann. Dieser Precompiler ersetzt die eingebetteten SQL-Anweisungen (`EXEC SQL`) durch Funktionsaufrufe der DBMS-Laufzeitbibliothek. Damit muss für jedes zu verwendende DBMS ein spezifischer Precompiler eingesetzt werden.

© Thorsten Strobel, 2002

1.2 ODBC

Der gleiche Anwendungsfall wie in Abbildung 1 soll nun mit Hilfe der ODBC-API in C realisiert werden (siehe Abbildung 2). Auch bei diesem Codebeispiel wird auf die Darstellung des Aufbaus und Abbaus der Verbindung zum DBMS verzichtet.

```
1  #include <sql.h> /* enthält alle ODBC-Funktionen und Konstanten    */
2
3  /* Deklaration der Hostvariablen. Über sie erfolgt der Austausch   */
4  /* der Daten zwischen Datenbankoperation und C-Programm            */
5  RETCODE rc;         /* Rückgabewert für ODBC-Funktionen            */
6  HSTMT hstmt;        /* Anweisungs-Handle, muss einer bestimmten    */
7                      /* Verbindung zum DBMS zugeordnet werden (hier */
8                      /* nicht weiter dargestellt)                   */
9  char NAME[26];
10 short TNUM;
11
12 main()
13 {
14 /* Abfrage der Zeilen, die mit einer best. Tel.nummer übereinstimmen  */
15 /* rc stellt einen Zeiger auf die aus der Abfrage resultierenden      */
16 /* Zeilen dar. Er wird in einer while-Schleife zeilenweise ausgelesen */
17 SQLExecuteDirect(hstmt, “select empname, telno from emp where telno
   >7301“);
18
19 for (rc = SQLFetch(hstmt); rc == SQL_SUCCESS; rc = SQLFetch(hstmt))
20     {
21     SQLGetData(hstmt, 1, NAME)
22     SQLGetData(hstmt, 2, TNUM)
23     }
```

Abbildung 2 Codebeispiel ODBC

ODBC bildet eine Anwendungsschnittstelle, die eine Menge von Funktionen (z.B. *SQLConnect*, *SQLExecuteDirect*, ...) zur Verfügung stellt. Da diese Funktionen in ODBC standardisiert sind und damit nicht vom DBMS abhängen, braucht die Anwendung nur einmal übersetzt werden und kann dann auf verschiedene DBMS zugreifen.

1.3 4GL

Bei Sprachen der vierten Generation (4GL) bzw. RAD-Werkzeugen (Rapid Application Programming Tool, z.B. Sybase PowerBuilder mit 4GL PowerScript) werden die Programmier-Paradigmen in Aufrufe der ODBC-API abgebildet (siehe Abbildung 3).

```
1  /* Deklaration der Hostvariablen. Über sie erfolgt der Austausch */
2  /* der Daten zwischen Datenbankoperation und PowerScript-Prog.   */
3  string NAME;
4  int TNUM;
5
6  /* Abfrage der Zeilen, die mit einer best. Tel.nummer übereinstimmen */
7  SELECT EMPNAME, TELNO INTO :NAME, :TNUM FROM EMP WHERE TELNO = 7301;
```

Abbildung 3 Codebeispiel PowerScript

2 Aufbau und Funktion von ODBC

Eine Datenbank-Anwendung, welche über ODBC auf eine Datenquelle zugreift besteht aus vier Komponenten: Anwendung, ODBC-Treiber-Manager, ODBC-Treiber und Datenquelle (siehe Abbildung 4).

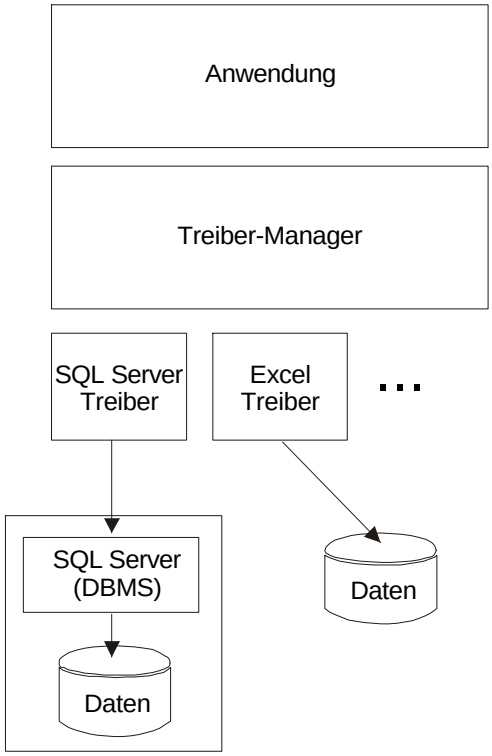

Abbildung 4 ODBC-Architektur

2.1 Datenquelle

Die Datenquelle enthält die für die Anwendung relevanten Daten. Dabei kann die Datenquelle aus einer Datenbankdatei (Textdatei, Exceldatei) oder einer Datenbankdatei und DBMS für die Zugriffsverwaltung bestehen (in diesem Fall nennt man Datenbankdatei und DBMS zusammen Datenbanksystem).

2.2 Anwendung

Eine Anwendung stößt durch einen Befehl einen Verbindungsaufbau zu einer ODBC-Datenquelle an. In diesem Befehl ist die adressierte ODBC-Datenquelle in Form des so genannten *Data Source Name* (DSN, Datenquellenname) enthalten. Unter dem DSN ist eine bestimmte Datenquelle im ODBC-Treiber-Manager registriert. Der Befehl zum Verbindungsaufbau kann weitere zum Verbindungsaufbau notwendige Parameter wie z.B. Benutzer-Kennung und Passwort enthalten. Ist die Verbindung aufgebaut, so kann die Anwendung SQL-Anweisungen auf die Datenquelle wie in 1.2 gezeigt ausführen.

2.3 ODBC-Treiber-Manager

Beim Treiber-Manager handelt es sich um eine DLL, die von Microsoft als Teil der ODBC-Installation auf dem Rechner der Datenbank-Anwendung bereitgestellt wird. Die Aufgabe des Treiber-Managers ist es, die jeweils benötigten ODBC-Treiber zu laden. Der Treiber-Manager verwaltet die Liste der auf dem Rechner vorhandenen DSN in einer Initialisierungsdatei (ODBCINST.INI) oder der Registry. Die DSN können dabei u.a. entweder als benutzerspezifische (d.h. nur für den jeweiligen Benutzer verfügbare) DSN oder systemweite (d.h. einem Computer zugewiesene DSN, ab ODBC 2.5) angelegt werden.

2.4 ODBC-Treiber

Auch beim ODBC-Treiber handelt es sich um eine DLL. Die für den jeweiligen Treiber notwendigen Parameter zum Verbindungsaufbau mit einer Datenquelle werden ebenfalls in einer Initialisierungsdatei (ODBC.INI) bzw. in der Registry gespeichert. Ein ODBC-Treiber enthält Funktionen zum Verbindungsaufbau mit einer Datenquelle, er interpretiert die Abfragen der Anwendung und sendet sie an die Datenquelle. Außerdem übernimmt er Formatkonvertierungen (z.B. Zeichensätze) zwischen Anwendung und Datenquelle. Nachdem die Ergebnisse aus der Datenquelle ermittelt sind, gibt der Treiber sie an die Anwendung zurück. Fehlermeldungen aus der Datenquelle werden vom ODBC-Treiber interpretiert und in Form eines Standard-Fehlercodes an die Anwendung zurückgegeben.

Es exisitieren zwei Arten von ODBC-Treibern:

- **Single-Tier-Treiber** stellen zum einen eine Verbindung zu einer Datenquelle her. Zum anderen enthalten sie die komplette Funktionalität, um direkt eine Datenbankdatei anzusprechen. Dies ist zum Beispiel notwendig, um Textdateien oder Exceldateien als Datenquelle in SQL-Anweisungen benutzen zu können. Dabei müssen diese Dateien ein bestimmtes Format aufweisen (z.B. bei Textdateien: Verzeichnis stellt Datenquelle dar, Datei stellt eine Tabelle dar innerhalb derer die Werte der Spalten z.B. durch Komma getrennt stehen; bei Excel: Datei stellt Datenquelle dar, Excel-Tabelle stellt Datenbank-Tabelle dar.). Der Single-Tier-Treiber setzt Datenbankabfragen (SQL-Anweisungen) aus der Anwendung in Abfragen an die Datenquelle um und agiert damit quasi als DBMS, indem er die Datenbankabfragen selbst verarbeitet. Dabei benutzt er z.T. Bestandteile (DLLs) der Datenquellen-Anwendung (z.B. Excel). Ein Single-Tier-Treiber ist in Abbildung 4 als Excel-Treiber dargestellt.
- **Multi-Tier-Treiber** senden Datenbankabfragen an einen Datenbank-Server, der diese verarbeitet. Eventuell werden die Datenbankabfragen dabei in ein für den Server verständliches Format umgewandelt. Dabei kann sich der Server entweder auf dem lokalen Rechner oder auf einem anderen Rechner in einem Netzwerk befinden. Ein Multi-Tier-Treiber ist in Abbildung 4 als SQL Server Treiber dargestellt.

Ein ODBC-Treiber kann somit Folgendes beinhalten:

- Eine Bibliothek mit Funktionsaufrufen zum Verbindungsaufbau (inkl. Anmeldung beim DBMS), zum Ausführen von SQL-Anweisungen und zur Ermittlung der Ergebnisse. Zum Verbindungsaufbau kann z.B. entweder die Funktion *SQLConnect*, *SQLDriverConnect* oder *SQLBrowseConnect* verwendet werden. Bei *SQLConnect* liefert die Anwendung die Verbindungsdaten (Namen der Datenquelle, die User-ID und Passwort) mit. Bei *SQLDriverConnect* können die Verbindungsdaten entweder durch die Anwendung mitgeliefert oder durch Dialogfenster des ODBC-Treiber-Managers (Auswahl der Datenquelle) und des ODBC-Treibers (User-ID, Passwort) abgefragt werden. *SQLBrowseConnect* ermittelt die Verbindungsattribute, welche der ODBC-Treiber zum Verbindungsaufbau benötigt, sodass die Anwendung darauf aufbauend ein eigenes Dialogfenster zum Verbindungsaufbau erzeugen kann.
- Eine standardisierte SQL-Syntax (eventuell in verschiedenen Ausbaustufen, Konformitätsebenen). Es kann eine SQL-Umwandlung vorgenommen werden, wenn die SQL-Syntax des DBMS nicht der standardisierten SQL-Syntax entspricht (z.B. Standard-Ungleichsoperator != aus der Anwendung wird bei Microsoft SQL Server durch DBMS-spezifischen Ungleichsoperator <> ersetzt).
- Eine Standardmenge an Fehlercodes und Fehlerfunktionen (z.B. zur Rückgabe des Fehlercodes und der Fehlermeldung des DBMS an die Anwendung).
- Eine Standardmethode zur Konvertierung von Datentypen und Zeichensätzen.
- Katalogfunktionen: standardisierte Informationen über den Aufbau der Datenbank (Systemtabellen)

3 Literatur

[1] Richard J. Simon: *Multimedia, ODBC und Telefonie*, SAMS, Haar bei München, 1997, ISBN 3-8272-4500-1.

[2] Kyle Geiger: *Inside ODBC*, Microsoft Press Deutschland, 1995, ISBN 3-86063-359-7.

[3] *DataWindow Builder Connection Reference*, Sybase, 1998.