

Probleme von Datenbank-Schnittstellen

Aufgabe von Datenbank-Schnittstellen:

Abbildung der unterschiedlichen Datenmodelle und Zugriffsparadigmen zwischen Programmiersprache und dem DBMS → **Impedance Mismatch**

Programmiersprache / C	DBMS / SQL
<ul style="list-style-type: none"> • Imperative Programmiersprache (wie kommt man zum Ergebnis) • Basisdatentypen und flexible Typkonstrukturen wie Strukturen und Klassen • Pointer • Arrays ➤ Insbesondere: Keine Mengen 	<ul style="list-style-type: none"> • Deklarative Anfragesprache (was soll das Ergebnis sein) • Plattform- und Programmiersprachen unabhängige Basisdatentypen • Arbeit mit Mengen • Relationen und Attribute ➤ Insbesondere: Keine Pointer, Schleifen, Verzweigungen, komplexen Strukturen

Datentransfer zwischen beiden Modellen ist also schwierig

→ **Maßnahmen zur Überwindung der sog. „Fehlanpassung“ erforderlich**

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

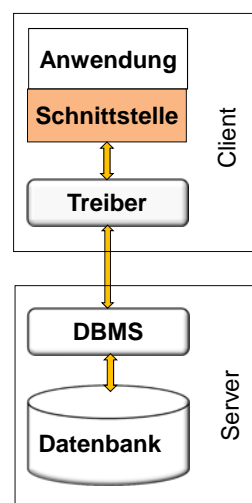
Datenbanksysteme II
Erweiterungen von Datenbanksprachen

Folie 1.1



Aufgaben von Schnittstellen zwischen Datenbank-Server und Client-Anwendung

- **Kapselung der Datenbankfunktionalität** durch geeignete Funktionen/Strukturen/Klassen für
 - Verbindung zum DBMS
 - Zugriff auf Konkrete Datenbank
 - Absetzen von Anfragen
 - ...
- **Zugriff auf Ergebnisse**
 - Geeignete Datenstrukturen für mengenwertige Anfrageergebnisse
 - Zugriff über imperative Programmiersprache → Cursor- oder Iterator-Konzept zum zeilenweisen Auslesen der Ergebnisse
 - Zugriff auf Metadaten (Beschreibung von Tabellen und Anfrageergebnissen, z.B. welche Spalten hat das gerade übertragene Ergebnis)
- Umgesetzt als Bibliotheken, die auf Treiber (optional) und Protokoll zur Kommunikation mit DBMS Server abbilden



Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Erweiterungen von Datenbanksprachen

Folie 1.2



Auswahl an Schnittstellen zwischen Datenbank-Server und Client-Anwendung

1. Embedded SQL (ESQL)

ANSI-Standard-API

2. Call-Level Schnittstellen (CLI)

SQL-Anweisungen gehen direkt an DB; setzen von Zeigern auf Ein- und Ausgabepuffer, Nutzung von Pointer

- ODBC – Open Database Connectivity
- DB-Library (Sybase / MS SQL 2000)
- Oracle Call Interface (OCI)

3. Objektorientierte Schnittstellen

- ASP.NET, OLE DB (MS) ⇒ C, C++, C#
- ActiveX Data Objects (ADO) ⇒ Visual Basic
- Remote Data Objects (RDO) ⇒ ODBC
- Document Object Model (DOM) ⇒ API für HTML-u. XML-Doc.
- Extensible Markup Language (XML)/ SGML (Standard Generalized ML)

4. Java-basierte Datenbankschnittstellen

- Java Database Connectivity (JDBC)

5. Scriptbasierte Schnittstellen

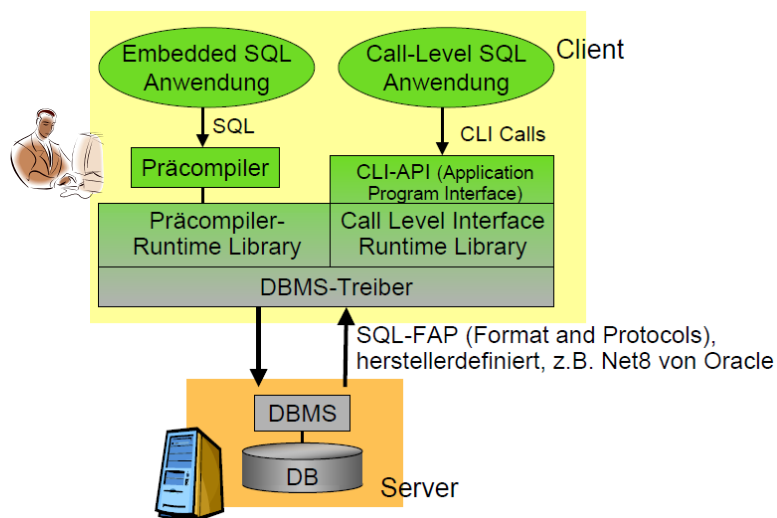
- Common Gateway Interface (CGI)
- Personal Home Page (PHP) / Perl
- Java-Script

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Erweiterungen von Datenbanksprachen

Folie 1.3 

Architekturvarianten für Datenbank-Schnittstellen



Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Erweiterungen von Datenbanksprachen

Folie 1.4 

Call-Level Interface (CLI)

- **Standardisierte Schnittstelle zur Kommunikation mit Datenbanken ohne Präcompiler**
 - CLI-API erlaubt das Erstellen und Ausführen von SQL-Anweisungen zur Laufzeit
- **SQL/CLI: Standard seit 1996**
 - Ursprung in Aktivitäten der SQL Access Group (SAG) mit dem Ziel, einen vereinheitlichten Zugriff auf Datenbanken bereitzustellen
 - Ist auch Bestandteil von SQL3
- **Standard-CLI umfasst über 40 Routinen:**
 - Verbindungskontrolle
 - Ressourcen-Allokation
 - Ausführung von SQL-Befehlen
 - Zugriff auf Diagnoseinformation
 - Transaktionsklammerung, Informationsanforderung zur Implementierung
- **CLI-Call wird durch den proprietären DBMS-Treiber umgesetzt in herstellerspezifischen DB-Zugriff**
 - Mit geeignetem Treiber kann prinzipiell jede Datenquelle angesprochen werden

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Erweiterungen von Datenbanksprachen

Folie 1.5

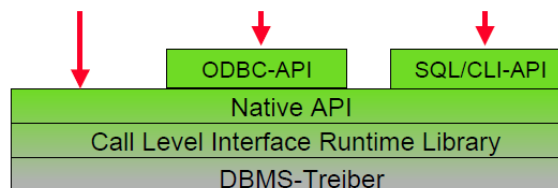


Call-Level Interface-Dialekte

Neben standardisiertem SQL/CLI haben sich weitere Dialekte herausgebildet
(war das ursprüngliche Ziel der SAG nicht die Definition einer einheitlichen Schnittstelle?)

- **ODBC (Open Database Connectivity):** Microsofts CLI-Variante
 - Beinhaltet die meisten (nicht alle) API-Calls des CLI-Standards
 - Zusätzlich: spezielle Calls zur Unterstützung von Microsoft-Programmen (Access, Excel, etc.)
- **JDBC / JDK** (Zugang von Java zu relationalen Datenbanken)
- **Proprietäre CLIs der DB-Hersteller ("Native API")** z.B.
 - Oracle Call Interface (OCI)
 - Sybase Open Client

In der Regel werden mehrere CLI-Dialekte parallel unterstützt



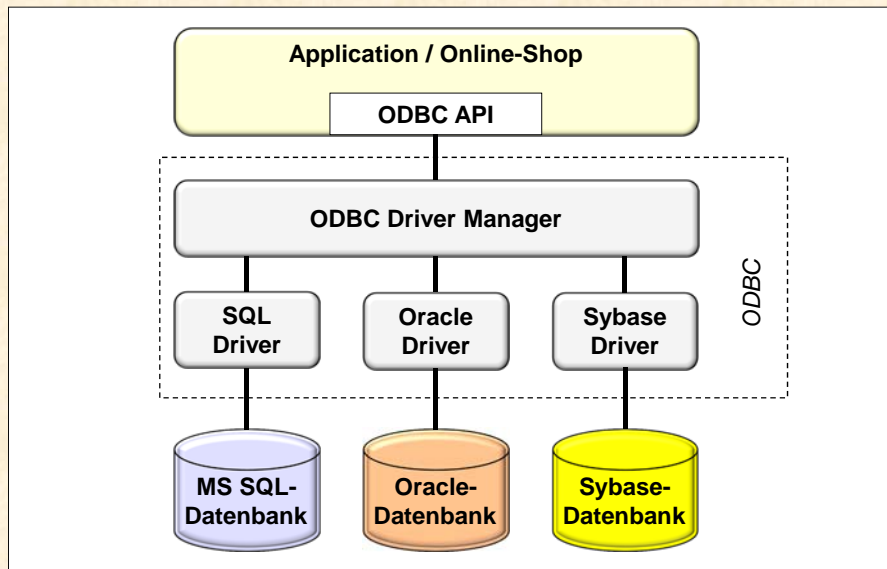
Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Erweiterungen von Datenbanksprachen

Folie 1.6



Architektur der ODBC-Schnittstelle



Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Erweiterungen von Datenbanksprachen

Folie 1.7

Merkmale von ODBC

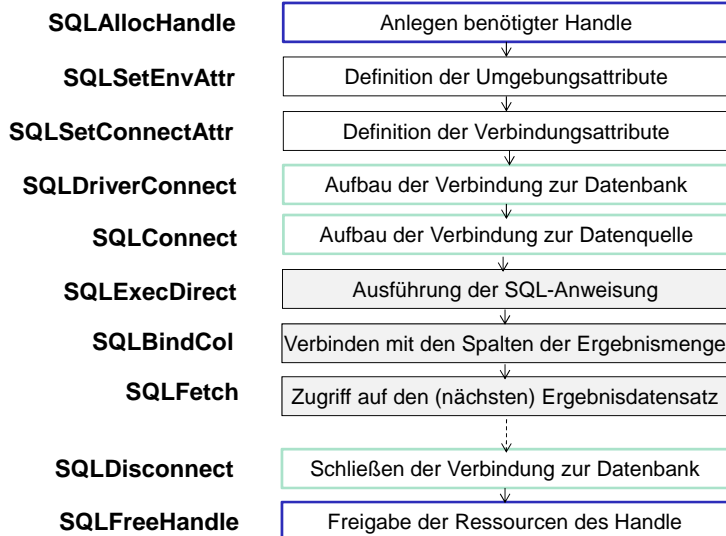
- **Low-level-Interface:**
Aufbau von Verbindungen, Absetzen von Anfragen, Lesen generischer Ergebnisse (im Gegensatz zu High-level-Interface mit z.B. definierter/definierbarer Abbildung auf Anwendungsobjekte)
- **Unabhängig von Programmiersprache:**
Schnittstelle besteht aus Funktionen mit Handles (Strukturen) zur Verwaltung der Zustandsinformationen
- **Unabhängig von Hardware und Betriebssystem:**
Ursprünglich Umsetzung des CLI-Standards (Call Level Interface) für Microsoft Windows, mittlerweile aber auf fast allen Plattformen
- **Unabhängig vom verwendeten DBMS:**
Treiber für fast alle kommerziellen DBMS verfügbar
- **Extrem flexibel,**
dafür aber nicht sehr einfach in der Handhabung

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Erweiterungen von Datenbanksprachen

Folie 1.8

Aufbau einer Datenbank-Verbindung über ODBC



Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Erweiterungen von Datenbanksprachen

Folie 1.9 

Programmausschnitt ODBC

```


...
char* selectStmt = "SELECT Kunr, Name, Kredit FROM Kunde WHERE Ort='Dresden' ";

retcode = SQLExecDirect(hstmt, (SQLCHAR *)selectStmt, SQL_NTS);
retcode = SQLBindCol(hstmt, 1, SQL_C_CHAR, szKunr, NR_LEN, &cbKunr);
retcode = SQLBindCol(hstmt, 2, SQL_C_CHAR, szName, NAME_LEN, &cbName);
retcode = SQLBindCol(hstmt, 3, SQL_C_DECIMAL, &szKredit, Kred_LEN, &cbKred);
//Pointer
for (int i=0; i++) {
    retcode = SQLFetch(hstmt);
    if (retcode == SQL_ERROR || retcode == SQL_SUCCESS_WITH_INFO)
        printf("SQL_ERROR\n");
    if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
        printf("%d: %s %s %f\n", i + 1, szKunr, szName, szKredit);
    else
        break;
}
...

```

Prof. Dr. oec. G. Gräfe
Prof. Dr.-Ing. A. Toll

Datenbanksysteme II
Erweiterungen von Datenbanksprachen

Folie 1.10 

Ergebnisdatentypen ODBC

- Ergebnis- bzw. Verbunddatentypen werden für die Kommunikation zwischen Client-Anwendung und Datenbank beim Aufruf des API-Calls **SQLBindCol** benötigt
- Ergebnis- bzw. Verbunddatentypen sind in **sqltypes.h** und **sql.h** definiert

Ausgewählte Ergebnis- bzw. Verbunddatentypen für C:

Datentyp im SQL-Server	ODBC-C-Verbunddatentyp	ODBC-C-Buffer	C-Typ (ab Vers. 3.0)
CHAR(n) VARCHAR(n)	SQL_C_CHAR; SQL_C_CHAR;	SQLCHAR SQLCHAR	char* char*
INTEGER SMALLINT FLOAT	SQL_C_LONG; SQL_C_SHORT; SQL_C_DOUBLE;	SQLINTEGER SQLSMALLINT SQLDOUBLE	long int short int double
DATE	SQL_C_TIMESTAMP;	SQL_TIMESTAMP_STRUCT (year, month, day);	struct