5. DB-Anwendungsprogrammierung

Wunsch:
Zugriff auf Datenbank aus Anwendungsprogramm

Probleme:

- (rel.) Datenbanksysteme arbeiten mengenorientiert, d.h. eine Anfrage liefert i.d.R. eine Tupelmenge als Ergebnis

  $\rightarrow$ deklarative Formulierung von Anfragen

- tupel- (datensatz-) weise Verarbeitung in den üblichen Programmiersprachen

  $\rightarrow$ operationale Vorgehensweise

$\Longrightarrow$ *impedance mismatch*

Anbindung von SQL

- *prozedurale Schnittstelle* oder **call**-*Schnittstelle* (SQL/CLI, ODBC, JDBC)

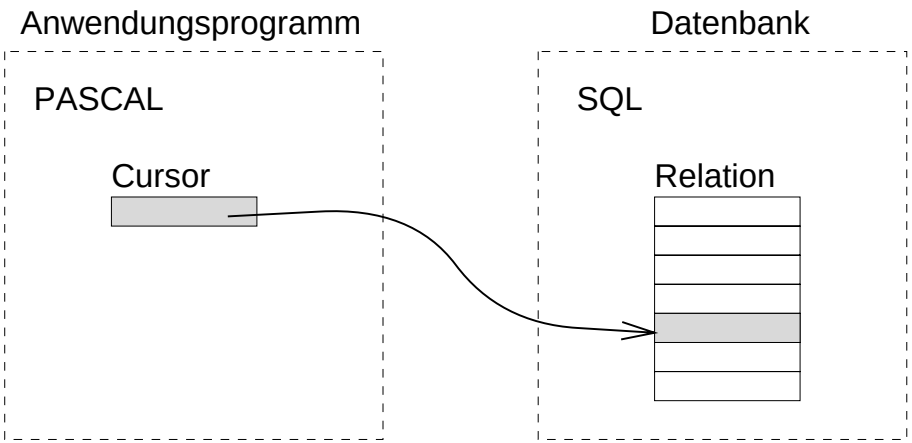
- *Einbettung* von Datenbanksprache in Programmiersprachen: Embedded SQL
  - statische Einbettung: *Vorübersetzer-Prinzip*

    $\rightsquigarrow$ SQL-Anweisungen *zur Übersetzungszeit* festgelegt
  - dynamische Einbettung (Dynamic SQL):

    $\rightsquigarrow$ Konstruktion von SQL-Anweisungen zur Laufzeit

- *Spracherweiterungen* und neue *Sprachentwicklungen*

Das Cursor-Konzept



Cursor-Deklaration:

```
declare AktBuch cursor for
select ISBN, Titel, Verlagsname
from Bücher
where Verlagsname = 'Thomson';
```

Cursor-Deklaration mit Änderungsmöglichkeit:

```
declare AktBuch cursor for
select ISBN, Titel, Verlagsname
from Bücher
for update of ISBN, Titel;
```


Cursor in SQL2

declare `CursorName`
[insensitive] [scroll] cursor for ...

- **insensitive**: Cursor verhält sich bei zwischenzeitlichen Änderungen am Datenbestand so, als ob er auf einer Kopie der ursprünglichen Daten arbeitet.
- **scroll**: erlaubt in der **fetch**-Anweisung beliebiges Navigieren in der Ergebnismenge;

  fetch kann dann mit folgenden Angaben durchgeführt werden:
  - **next**: Gehe weiter zum nächsten Tupel (default-Einstellung).
  - **prior**: Gehe zum vorherigen Tupel.
  - **first** bzw. **last**: Gehe zum ersten bzw. letzten Tupel.
  - **absolute** n **from**: Gehe zum n-ten Tupel des Cursors. Negative Werte werden relativ zum letzten Tupel rückwärts gewertet — **absolute** -1 ist also äquivalent zu **last**.
  - **relative** n **from**: Gehe zum n-ten Tupel relativ zur aktuellen Cursor-Position.

Statische Einbettung: Embedded SQL

```
exec sql declare AktBuch cursor for
         select ISBN, Titel, Verlagsname
         from Bücher
         for update of ISBN, Titel;
```

Schlüsselwort **exec** dient zur Erkennung von Embedded SQL bei der Compilierung von Anwendungsprogrammen.

Öffnen und Schließen einer Datenbank:

```
exec sql connect UniBeispiel;
```

Deklaration benutzter Datenbankrelationen:

```
exec sql declare Buch table
         ( ISBN char(10) not null,
           Titel char(120) not null,
           Verlagsname char(30) not null);
```


Deklaration gemeinsamer Variablen

```
exec sql begin declare section;
         BuchISBN char(10);
         NeuerPreis real;
exec sql end declare section;
```

Benutzung deklarierter Variablen in SQL:

```
exec sql update Buch_Versionen
         set Preis = :NeuerPreis
         where ISBN = :BuchISDN ;
```

```
exec sql insert into  Buch_Versionen
         values (:NeuISBN, :NeuAuflage, 1995,
                 :Seiten, :Preis);
```

Datentransfer zwischen Datenbank und Programm

```
exec sql select ISBN, Auflage, Jahr, Seiten, Preis
         into   :ISBN, :Auflage, :Jahr, :Seiten, :Preis
         from   Buch_Versionen
         where ISBN = :SuchISBN and Auflage = 1;
```

Indikator-Variablen zum Test auf **null**-Werte (da der Wert **null** in Programmiersprachen nicht zur Verfügung steht):

```
exec sql select :ISBN, Auflage, Jahr, Seiten, Preis
         into   :ISBN, :Auflage, :Jahr,
                :Seiten:SeitenInd, :Preis:PreisInd
         from   Buch_Versionen
         where ISBN = :SuchISBN and
               Auflage = :SuchAuflage;
```


Einsatz der Cursor-Technik

- Öffnen des Cursor:

```
exec sql open AktBuch;
```

- Holen eines Tupels:

```
exec sql fetch AktBuch
         into :ISBN, :Titel, :Verlagsname;
```

- Löschen des aktuellen Tupels:

```
exec sql delete
         from Bücher
         where current of AktBuch;
```

- Schließen des Cursor:

```
exec sql close AktBuch;
```

Fehler- und Ausnahmebehandlung

SQL Communication Area

exec sql include sqlca;

'**whenever**'-Anweisung:

exec sql whenever <Bedingung> <Aktion>;

- **not found**: Kein Tupel wurde gefunden, definiert etwa als $\mathbf{sqlcode} = 100$.
- **sqlwarning**: Warnung, entspricht etwa $\mathbf{sqlcode} > 0 \wedge \mathbf{sqlcode} \neq 100$.
- **sqlerror**: Fehler, also $\mathbf{sqlcode} < 0$.

Transaktionssteuerung:

exec sql commit work;

exec sql rollback work;

Ein Beispielprogramm

```
...
type Binlist = ...;
procedure InitList (Binlist):  ...;
procedure AddToList (string, string, Binlist):  ...;
procedure TransClos (Binlist, Binlist):  ...;
var Eingabe, Ausgabe:  Binlist;

exec sql declare Voraussetzung table
   ( IstVoraus char(30) not null,
     Fuer char(30) not null);

exec sql begin declare section;
   IstVoraus char(30);
   Fuer char(30);
exec sql end declare section;

exec sql declare AktVor cursor for
   select * from Voraussetzung;

begin
   InitList(Eingabe);
   exec sql open AktVor;
   exec sql whenever not found goto Weiter;
   loop
      exec sql fetch AktVor into :IstVoraus, :Von;
      AddToList(IstVoraus, Von, Eingabe);
   end loop;
Weiter:
   exec sql close AktVor;
   TransClos(Eingabe, Ausgabe);
   .../* Ausgabe der berechneten transitiven Hülle */
end.
```

Dynamische Einbettung: Dynamic SQL

```
exec sql begin declare section;
         dcl AnfrageString char(256) varying;
exec sql end declare section;
exec sql declare AnfrageObjekt statement;
AnfrageString :=
'DELETE FROM Vorlesungen WHERE SWS < 2';
...
exec sql prepare AnfrageObjekt
         from    :AnfrageString;
exec sql execute AnfrageObjekt;
```

$\rightsquigarrow$ „Anfragen als Zeichenketten“

```
...
AnfrageString :=
   'DELETE FROM Buch_Versionen ' +
   'WHERE ISBN = ?  AND Auflage = ?'  ;
exec sql prepare AnfrageObjekt
         from    :AnfrageString;
exec sql execute AnfrageObjekt
         using :LöschISBN, :LöschAuflage;
```

$\rightarrow$ Wertübergabe an Anfragen

Weitere Ansätze

- Prozedurale SQL-Erweiterungen
- Gespeicherte Prozeduren
- 4GL: Sprachen der vierten Generation
- Datenbankprogrammiersprachen
- Persistente (objektorientierte) Programmiersprachen