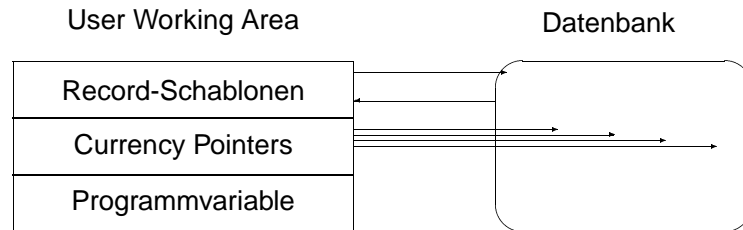


Datenmanipulation im Netzwerkmodell

User Working Area UWA



Navigieren mit dem find-Befehl

1. Mittels des **find**-Befehls wird der gesuchte Record lokalisiert und zum **current of run-unit**.
2. Das **get**-Kommando kopiert den unter **current of run-unit** stehenden Record in die passende Record-Schablone der UWA.
3. Zugriff auf Hauptprogramm auf UWA-Schablonen

Datenbank-Anwendungsprogrammierung

- ➡ Navigierende Ansätze
- ➡ Anbindung von SQL
- ➡ Weitere Ansätze

Zeiger im UWA

- *Current of run-unit*: Letzter Record, auf den im Programm zugegriffen wurde.
- *Current of record type*: Für jeden Record Type T wird auf den zuletzt zugegriffenen Record mit **current of T** verwiesen.
- *Current of set type*: Für jeden Set-Typ S wird auf den zuletzt zugegriffenen Record (owner oder member) mit **current of S** verwiesen.

Arten der find-Anweisung II

- Durchlauf durch Set Occurrence:

find owner of current *x* **set** ;
find next *y* **record in current** *x* **set**

Angabe **owner is system**: sequentieller Durchlauf über alle Elemente.

- In Set-Ausprägung: Suche nach Attributwerten
- Finden des Owners eines Records.
- **current of** *T* wird **current of run-unit**.

Einfügen im NWM

Bei **insertion is automatic**:

- Einfügen an der Position des aktuellen *Currency Pointers* des Set-Typs:

set selection is thru current of *x* **set**

- Der Owner der Set-Ausprägung wird anhand eines berechneten **calc**-Schlüssels bestimmt:

set selection is thru owner
using Feldliste für **calc**-Schlüssel

Arten der find-Anweisung

- Direktzugriff über Schlüssel:

find *x* **record by database key** *y*

- Analog über **calc**-Schlüssel (berechneter Hash-Wert):

find *x* **record by calc-key**

- Für gegebenen **calc**-Schlüssel alle Records dazu:

find duplicate *x* **record by calc-key**

Datenmanipulation im NWM

- Das Einfügen wird als **store** bezeichnet.

- ◆ Die **store**-Operation gibt es für Record-Typen wie auch für Set-Ausprägungen.
- ◆ Ein **store** für einen Record-Typ ist ein Transfer der Wertebelegung der entsprechenden Record-Schablone vom Anwendungsprogramm in das Datenbanknetzwerk.

- Das Löschen wird für Record-Typen als **delete** bezeichnet. Das Herausnehmen aus einer Set-Ausprägung wird als **remove** bezeichnet.

- Ändern von Attributen erfolgt mittels **modify**.

Datenmanipulation im hierarchischen Modell

get-Kommando zum Navigieren innerhalb der hierarchisch angeordneten Datensätze:

- (1) **get unique** x [**where** Bedingungen];
- (2) **get next** x [**where** Bedingungen];
- (3) **get next within parent**

Anbindung von SQL

- *prozedurale Schnittstelle* oder **call-Schnittstelle** (SQL/CLI, ODBC, JDBC)
- *Einbettung* von Datenbanksprache in Programmiersprachen: Embedded SQL
 - ◆ statische Einbettung: *Vorübersetzer-Prinzip*
~> SQL-Anweisungen zur *Übersetzungszeit* festgelegt.
 - ◆ dynamische Einbettung:
~> Konstruktion von SQL-Anweisungen zur Laufzeit
- *Spracherweiterungen* und neue *Sprachentwicklungen*

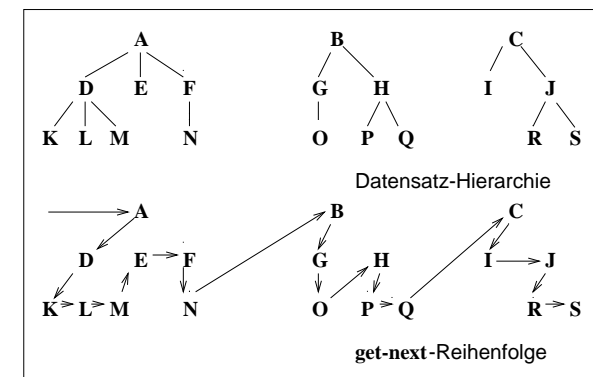
Einfügen im NWM II

Bei **insertion is manual**, muß die Einfügeposition durch die *Cursor Pointer* explizit bestimmt werden:

insert x **into** y

Hierbei ist x ein Record-Typ und y ein Set-Typ.

Abarbeitungsreihenfolge im HM



Cursor in SQL

Cursor-Deklaration:

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

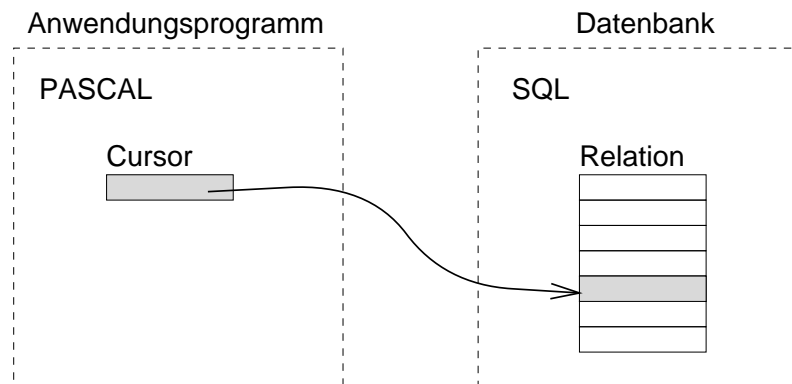
Cursor-Deklaration mit Änderungsmöglichkeit:

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

Statische Einbettung: Embedded SQL

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

Das Cursor-Konzept



Cursor in SQL2

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

- **next**: Gehe weiter zum nächsten Tupel (wie bisher).
- **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.

Deklaration gemeinsamer Variablen

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

Benutzung deklarerter Variablen in SQL:

```
exec sql update Buch_Versionen  
    set Preis = :NeuerPreis where ISBN = :BuchISBN ;  
  
exec sql insert into Buch_Versionen  
    values ( :NeuISBN, :NeuAuflage, 1995,  
            :Seiten, :Preis);
```

Einsatz der Cursor-Technik

```
exec sql open AktBuch;  
  
exec sql fetch AktBuch  
    into :ISBN, :Titel, :Verlagsname;  
  
exec sql close AktBuch;  
  
exec sql delete  
    from Bücher  
    where current of AktBuch;
```

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);
```

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:

```
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;
```

Transaktionssteuerung

```
exec sql commit work;
```

```
exec sql rollback work;
```

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;
```

“Anfragen als Zeichenketten”

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 `sqlcode = 100`.
- **sqlwarning**: Warnung, entspricht etwa `sqlcode > 0 ∧ sqlcode ≠ 100`.
- **sqlerror**: Fehler, also `sqlcode < 0`.

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.
```

Weitere Ansätze

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

Prozedurale SQL-Erweiterungen: PL/SQL II

Cursor

```
cursor AktBuch is
  select ISBN, Titel, Verlagsname
  from Bücher;
```

Zugriff auf Typinformation

```
AktuellerPersonenName Mitarbeiter.PersonName%type;
BuchTupel               AktBuch%rowtype;
```

Dynamische Einbettung: Dynamic SQL

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

Wertübergabe an Anfragen

Prozedurale SQL-Erweiterungen: PL/SQL

Deklarationen

```
declare
  Heute date;
  type PersonRecordType is record
    ( PersonName varchar (50),
      GebDatum date);
  Mitarbeiter PersonRecordType;
```

```

if <Bedingung> then
  <PL/SQL-Anweisungen>
  [ else
    <PL/SQL-Anweisungen> ]
end if;

for <IndexVariable> in <EndlicherBereich>
loop
  <PL/SQL-Anweisungen>;
end loop;

while <Bedingung>
loop
  <PL/SQL-Anweisungen>;
end loop;

```

Beispiel in PL/SQL

```

declare
  AnzahlNeu number;
  AnzahlAlt number;
  fertig boolean;

begin

  create table TransHuelle
    ( IstVoraus char(30) not null,
      Fuer char(30) not null);

  insert into TransHuelle
  select * from Voraussetzungen;

  select count(*) into AnzahlNeu from TransHuelle;
  fertig := false;

  while not fertig
  loop
    AnzahlAlt := AnzahlNeu;
    insert into TransHuelle
    select T1.IstVoraus, T2.Fuer
    from TransHuelle T1, TransHuelle T2
    where T1.Fuer = T2.IstVoraus and
      not exists ( select * from TransHuelle T3
        where T3.IstVoraus = T1.IstVoraus
          and T3.Fuer = T2.Fuer);
    select count(*) into AnzahlNeu from TransHuelle;
    fertig = ( AnzahlAlt = AnzahlNeu );
  end loop;

```

```

for BuchRec in AktBuch
loop
  ...
end loop;

```

entspricht

```

declare
  ...
  BuchRec AktBuch%rowtype;
  ...
begin
  loop
    fetch AktBuch into BuchRec;
    exit when AktBuch%notfound;
    ...
  end loop;

```

Fehlerbehandlung

```

when Ausnahme then ProgrammStück;

```

“Stored Procedures”

```

create function FunktionsName
  ( Param1 ParamTyp1, ..., ParamN ParamTypN )
return ErgebnisTyp
is
  /* PL/SQL - Block mit return-Anweisung */

  create procedure ProzedurName
  ( Param1 in ParamTyp1,
    Param2 out ParamTyp2,
    Param3 in out ParamTyp3,
    ... )
is
  /* PL/SQL - Block mit Zuweisungen an out-Parameter */

```

Weitere prozedurale SQL-Erweiterungen

- SQL/PSM (SQL Persistent Stored Modules): SQL-Standard
- Transact-SQL: Sybase, Microsoft SQL Server
- externe Routinen: implementiert in C, Java, ...

INGRES Windows4GL

Programm besteht aus hierarchischen *Frames*

Frame besteht aus:

- Formular (Maske mit Eingabefeldern und Schaltern)
- Auslöserregeln
- Aktionen für Elemente des Formulars
 - ◆ Änderungsoperationen und Anfragen in SQL
 - ◆ Prozeduren mit **while**, **if**, **exit**, **callframe** (Aufruf eines Nachfolger-Frames), ...

Auslöserregeln und Aktionen: siehe aktive Datenbanken (nächstes Kapitel)

Vorteile von Gespeicherten Prozeduren

- Strukturierungsmittel für größere Anwendungen
- Prozeduren nur vom DBMS abhängig und nicht von externen Programmiersprachen oder Betriebssystemumgebungen
- *Optimierung* der Prozeduren
- Ausführung der Prozeduren unter Kontrolle des DBMS
- zentrale Kontrolle der Prozeduren: redundanzfreie Darstellung relevanter Aspekte der Anwendungsfunktionalität
- Rechtevergabe für Prozeduren
- in der Integritätssicherung: Aktionsteil von Triggern

4GL: Sprachen der vierten Generation

- 4GL: Fourth Generation Languages
- 3GL: Ada, Pascal, C, ... ; 5GL: PROLOG, ...
- 4GL-Sprachen für Anwendungen mit großen Datenmengen
- Sprachentwurf leider oft konzeptionslos
- SQL + imperative Konstrukte + ereignisgesteuerte Programmierung
- 4GL-Werkzeuge unterstützen interaktive, graphische Programmierung
 - ◆ Menüs und Masken (Formulare)
 - ◆ werden am Bildschirm "zusammengesetzt"
 - ◆ Auslöserregeln und Aktionen hinter Maskenelementen
- Beispiele
 - ◆ Klassiker: NATURAL, TOTAL
 - ◆ moderne 4GLs für RDBS: INGRES Windows4GL, Gupta SQL-Windows

Persistente Programmiersprachen

Persistenzkonzept von Atkinson: Objekte haben eine beliebige Lebensdauer unabhängig von der Blockstruktur des Programms, mit zwei Prinzipien:

- *Typ-Orthogonalität*: Programm-Objekte beliebigen Typs können persistent gemacht werden
- *Programm-Unabhängigkeit*: verbietet explizite **store**- oder **move**-Kommandos zum Speichern von Objekten

Neben persistenten Objekten auch *transiente Objekte*

Datenbankprogrammiersprachen

Problem: Impedance Mismatch (Kluft zwischen Typsystemen PL und DB)

- Datenbankprogrammiersprache (DBPL): DB-Typsystem in Programmiersprache
- Persistente Programmiersprache: DB-Persistenzkonzept für PL-Typsystem

DBPLs

- etwa J. Schmidt: Pascal/R, Modula/R, DBPL
- PL + persistenter Datentyp *relation*
- Operationen: Änderungsoperationen, Anfragen (etwa in Tupelkalkül-Notation, Iteratoren)

Realisierungstechniken für Persistenz

- *klassenabhängige Persistenz*: persistente Wurzelklasse impliziert Teilgraph aller persistenten Objekte
- *objektabhängige Persistenz*: jedes Objekt in jeder Klasse kann persistent oder transient sein

Persistenz-Fortpflanzung

- *explizite Persistenz*: jedes Objekt einzeln
- *Persistenz durch Erreichbarkeit*: Objekt persistent, dann automatisch auch (rekursiv) alle Komponentenobjekte persistent

ODMG-Binding: Persistenzkonzept für OOPL