Aufbau des Vortrages

- Neuerungen in SQL:1999
- Typsystem
- Client/Server Module
- SQL-invoked Routinen
- PSM

Neuerungen in SQL:1999

- neue Datentypen
- Rekursion (Breiten- und Tiefensuche)
- SQL Call-level Interface (CLI)
- Routinen in SQL
- Persistent Stored Modules (PSM)
- SQL Bindings
- Trigger

Typsystem

- Vordefinierte Typen:
  - Numerische Typen, Strings, BLOB, Boolean, Typen für Datum und Zeit, Intervalle
- Konstruierte atomare Typen
  - Refernce Type
- Konstruierte Zusammengesetzte Typen
  - Collection Types, Row Typ
- Benutzerdefinierte Typen
  - Distinct Type, Abstrakte Datentypen

Große Objekte (LOB)

- BLOB - Audiodaten, Bilder etc.
- CLOB - Große Zeichenmengen, Bücher, Lebenslauf etc.
- Spezifikation der Größe in KB, MB oder GB
- von Datenbank verwaltet und manipuliert
- Lokatoren (HOLD, FREE)

Große Objekte (2)

Unterstützte Operationen und Funktionen:

- Suchen, Ersetzen von Werten
- LIKE
- EQUAL/ NOT EQUAL
- TRIM, OVERLAY, LENGTH, POSITION
- Benutzerdefinierte Funktionen

Nicht unterstützte Operationen und Funktionen:

- GREATHER THAN, LESS THAN
- Primary oder foreign Keys, unique Constraint
- GROUP BY, ORDER BY
- Verbundattribut
- UNION, INTERSECT, EXCEPT

ARRAY

- jeder Datentyp zulässig (außer ARRAY)
- nur eine Dimension
- Deklarativer Zugriff oder über Indexangabe
- Lokatoren
- einziger Collection-Type in SQL:1999

ROW TYPE

- Folge von Attributnamen
- Verschachtelung möglich

```
CREATE TABLE Angestellte (
        Anr  INTEGER,
        Adresse  ROW  (
          Straße  CHAR (20),
                Stadt CHAR (20),
          Staat  CHAR (3)
        )
)
```

REFERENCE TYPE

- Einsetzbar wie andere Typen
- Repräsentation durch Benutzer oder System generiert
- Anwendungsbereich auf Tabelle beschränkbar
  - nur dann Dereferenz möglich

DISTINCT

- Trennung von logisch inkompatiblen Daten
- Transformation mit Hilfe der CAST Funktion
- Keine Vererbung

Abstrakte Datentypen (ADT)

- komplexe Strukturen
- Daten und Verhalten
- Zur Definition von Tabellen

Attribute in ADTs

- durch Namen und Typ festgelegt
- jeder Datentyp möglich
- Verschachtelung von ADTs
- Zugriff durch Punkt-Notation oder über Schnittstellenfunktionen

```
CREATE TYPE  Adresse AS (
        Straße   CHAR (30),
        Stadt    CHAR (30),
        Land     CHAR (3),
) NOT  FINAL;
CREATE TYPE  Person AS (
        Name  CHAR (30),
        Anschrift  Adresse,
        ....
)  NOT  FINAL;
```

Methoden in ADTs

- SQL-invoked Routinen, an ADT gebunden
- Muss in Typschema definiert werden
- spezifizieren Verhalten von ADTs
- Können überladen oder überschrieben werden
- Stellen Schnittstellen für Attribute dar

Initialisierung von ADTs

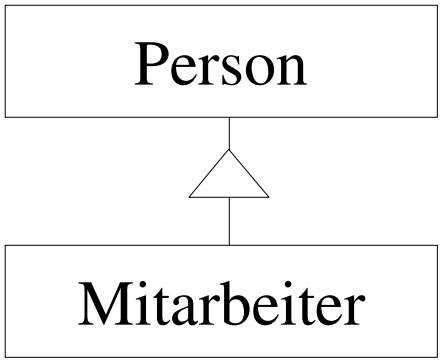
- Default Konstruktor
- Benutzerdefinierte Konstruktoren
  - Kann Parameter besitzen
  - Aufruf durch NEW Anweisung
  - Mehrere Konstruktoren möglich

```
BEGIN
   DECLARE  ad Adresse;
   SET  ad = NEW Adresse (...);
END
```

- INSTANTIABLE/ NOT INSTANTIABLE

Vererbung

- UNDER, INSTANTIABLE, FINAL Klausel
- Weiter Attribute/ Methoden können definiert werden
- Keine Mehrfachvererbung



Client Module

- existieren außerhalb des Datenbankkatalogs
- LANGUAGE Klausel
- SQL wird nicht unterstützt.
- beinhalten Prozeduren und Funktionen
- stellen Schnittstellen zu Datenbank dar
- beinhalten temporäre Tabellen

Server Module

- existieren im Server
- beinhalten Prozeduren und Funktionen
- beinhalten temporäre Tabellen

Schema Routinen

- existieren im Schema
- können Funktionen oder Prozeduren darstellen

SQL-invoked Routinen

- interne (SQL Routinen), externe Routinen
- Vorteile/Nachteile von SQL Routinen:
  - \+ integrieren Sprache und Umgebung (leichter zu benutzen)
  - \+ Keine Typumwandlung nötig
  - \- Nicht so umfangreich wie eine Hostsprache
- Vorteile/Nachteile von externen Routinen
  - \+ Komplette Programmiersprache
  - \+ Vorteile von Kode Bibliotheken
  - \- Konvertierung der Datentypen nötig
  - \- zwei Sprachen zu erlernen

Prozeduren, Funktionen, Methoden

- Prozeduren:
  - Input, Output und InOut Parameter
  - Kein Rückgabewert
- Funktionen:
  - Nur Input Parameter
  - Besitzen Rückgabewert
- Methoden:
  - Funktion, die an ADT gebunden ist
  - 2 Definitionsschritte: Signatur und Body separat
  - Muss in einem Typschema kreiert werden

Externe Routinen

- LANGUAGE Klausel
- SQL Data Access Indication Klausel
- PARAMETER STYLE SQL(Default-Wert) oder PARAMETER STYLE GENERAL
- DETERMINISTIC (Default-Wert) oder NOT DETERMINISTIC
- RETURNS NULL ON NULL INPUT oder CALLED ON NULL INPUT (Default-Wert)

SQL Routinen

- wird als Schemaobjekt gespeichert, Schemabesitzer hat EXECUTE Rechte
- Body kann sämtliche SQL Anweisungen beinhalten, außer CONNCET, DISCONNECT, COMMIT, ROLLBACK, dynamisches SQL oder DDL Anweisungen

Prozeduale Sprachelemente in SQL (1)

- Blockanweisung:
  - BEGIN ... END Anweisung
  - ATOMIC oder NOT ATOMIC (Default-Wert)
- Zuweisungen:
  - Ergebnis einer Zielvariablen zuordnen
  - SET Variablenname = Ausdruck;
- IF Anweisung:
  - Überprüfung von Bedingungen
  - IF Bedingung THEN Anweisung ELSEIF Bedingung THEN Anweisung..ELSE...END IF
  - ELSE IF und ELSE sind optional

Prozeduale Sprachelemente in SQL (2)

- CASE Anweisung:
  - Auswahl eines Anweisungsblockes
  - CASE Ausdruck WHEN Wert1 THEN Anweisung WHEN Wert2 THEN...ELSE..END CASE
  - ELSE ist optional
- FOR Anweisung:
  - Iteration über Ergebnistabelle
  - Anweisung für jede Zeile der Ergebnistabelle
  - Keine LEAVE Anweisung zulässig
  - Muss mit einem Suchausdruck verbunden sein
  - zu Beginn wird implizit ein Cursor geöffnet

```
DECLARE x INTEGER DEFAULT 0;
FOR L1 AS
SELECT Gehalt FROM Angestellte
DO
    SET x = x + Gehalt;
END FOR;
```

Prozeduale Sprachelemente in SQL (3)

- LOOP, WHILE, REPEAT und LEAVE Anweisung:
  - Möglichkeit zur Realisierung von Schleifen
  - LOOP Schleife wird durch LEAVE Anweisung beendet
  - LEAVE kann auch in anderen Schleifen oder Blöcken verwendet werden
- CONDITION HANDLER
  - Erlaubt auf Bedingungen während der Ausführung zu reagieren
  - Dient zur Realisierung von Systemverhalten, wie UNDO, REDO, CONTINUE, EXIT
  - Condition durch System oder Durch SIGNAL, RESIGNAL geworfen
  - Als CONTINUE, EXIT oder UNDO Handler definierbar
  - UNDO nur in ATOMIC Blöcken erlaubt

Zusammenfassung

- Erweiterung des Typsystems: Bessere Modellierung der Miniwelt
- Funktionen komplett in SQL
- Anwendungsprogramme leichter zu erstellen
- Vorteile des Objektorientierten Ansatzes (Vererbung)

Mögliche Neuerungen in SQL-4

- Mehrfachvererbung
- Möglichkeit zur Datenkapselung
- mehrdimensionale Arrays
- Weitere Collection Types, wie z.B. SET
- Datentypen und Operationen für multimedia Dateien