6.5.2 SQL-92

Erweiterungen bei Verbunden:

- explizites kartesisches Produkt (in Oracle: nur auf 1. Art):

  SELECT * FROM liefert, Artikel;

  oder

  SELECT * FROM liefert CROSS JOIN Artikel;

- natürlicher Verbund:
  SELECT * FROM Artikel, enthaelt
  WHERE Artikel.Artnr = enthaelt.Artnr;

  oder

  SELECT * FROM Artikel JOIN enthaelt
  ON Artikel.Artnr = enthaelt.Artnr;

  oder

  SELECT * FROM Artikel JOIN enthaelt
  USING (Artnr);

  oder

  SELECT * FROM Artikel NATURAL JOIN enthaelt;

- statt INNER auch OUTER JOIN (LEFT, RIGHT) (in Oracle mit (+)-Angabe rechts bzw. links, s. vorne)

- mit Tupelvariablen beim SELECT (nicht in Oracle):
  SELECT Ergebnis.Artnr
  FROM (Artikel NATURAL JOIN enthaelt);
  AS Ergebnis

SQL-92

- UNION an beliebiger Stelle (in Oracle: zwischen 2 bel. SFW-Statements)
- SELECT-Ausdrücke auch hinter FROM (in Oracle)
- INTERSECT und EXCEPT als Mengenoperationen für Durchschnitt und Differenz (in Oracle INTERSECT, MINUS)
- Angabe der Attribute bei Mengenoperationen (CORRESPONDING BY) (in Oracle: nur über alias AS...)
- Benennungen:
  SELECT COUNT(*) AS Anzahl..... (in Oracle)
- dreiwertige Logik (IS TRUE, IS FALSE, IS UNKNOWN) (in Oracle: FALSE, TRUE , UNKNOWN)
- Darstellung von Intervallen, Anfragen an Intervalle (OVERLAPS) (in Oracle nur: [NOT] IN, [NOT] BETWEEN x AND y)

6.5.3 SQL3

- enthält auch objektorientierte Features wie (Value- u. Object-)ADTs, ADT-Hierarchien, Einkapselung, Polymorphismus,...

- daneben relationale Erweiterungen wie Tabellen-Hierarchien, weitere Join-Typen (USING PRIMARY KEY/FOREIGN KEY/CONSTRAINT), rekursive Anfragen, Multimengen, Mengen u. Listen als komplexere Datentypen, Aufzählungstypen, Trigger, ...

- Produkte vermutlich erst in gut 5 Jahren erhältlich

SQL3

Beispiel:

```
CREATE OBJECT TYPE Person_adt
WITH OID NOT VISIBLE
(
PUBLIC Ausweisnr INTEGER PRIMARY KEY
PUBLIC Name VARCHAR2(30)
PRIVATE Gebdatum DATE
PUBLIC Alter INTEGER VIRTUAL
GET WITH Ermittle_Alter CHECK(VALUE.Alter > 0)
PUBLIC Vater Person_adt

PUBLIC ACTOR FUNCTION Ermittle_Alter (:p Person_adt)
RETURNS INTEGER;
RETURN (CURRENT_DATE - :p.Gebdatum)

PUBLIC ACTOR FUNKTION Einkommen (:p Person_adt)
RETURNS INTEGER;
RETURN 0

.....
);
```

```
CREATE OBJECT TYPE Student_adt
WITH OID NOT VISIBLE UNDER Person_adt
(
PUBLIC Matrnr INTEGER
PUBLIC Fach VARCHAR2(30)
PUBLIC Bafoeg INTEGER

PUBLIC ACTOR FUNKTION Einkommen (:p Student_adt)
RETURNS INTEGER;
RETURN :p.Bafoeg
);
```

```
CREATE TABLE Personen (
Person Person_adt INSTANCE
);
```

```
INSERT INTO Personen( Matrnr(Person), Name(Person), Geb-
datum(Person))
VALUES Person_adt(007,'James Bond', 05-Feb-45);
```