6.5.4 QUEL

- QUEry Language, von INGRES

- ist vollständige Datenbanksprache (IQL + DDL + VDL + DML + SSL), im folgenden nur IQL

- basiert auf Tupelrelationenkalkül:
  - alle Tupelvariablen sind $\exists$-quantifiziert, an Relationen gebunden
  - Bindung durch
    RANGE OF tupelvariable IS relationenname
  - Einschränkung einer Tupelvariablen t auf ein Attribut A: t.A

- Artikel={Artnr, EAN, Artname}
  liefert={LName, Artnr, Preis}
  Lieferant={LName, LAdresse}
  mit den Bindungen
  RANGE OF A IS Artikel
  RANGE OF A1 IS Artikel
  RANGE OF L IS liefert
  RANGE OF La IS Lieferant

- Syntax einer QUEL-Anfrage

  RETRIEVE (liste_von_tupelvariablen)
  WHERE bedingung

QUEL

Beispielanfragen:

- Was ist der Name des Artikels mit der Nummer 4711?
  RETRIEVE (A.Artname)
  WHERE A.Artnr = 4711

- Welche Namen und Preise haben Artikel, die von Osnabrücker Lieferanten geliefert werden?
  RETRIEVE (A.Artname, L.Preis)
  WHERE A.Artnr = L. Artnr
  AND L.LName = La.LName
  AND La.LAdresse = 'Osnabrueck'

- mit Umbenennung: Gibt es unterschiedliche Artikel mit dem gleichen Artikelnamen?
  RETRIEVE (T1 = A.Artnr, T2 = A1.Artnr)
  WHERE A.Artname = A1.Artname
  AND A.Artnr <> A1.Artnr

- Gib alle Attribute des Artikels 'Leitz-Ordner A4' aus!
  RETRIEVE A.ALL
  WHERE A.Artname = 'Leitz-Ordner A4'

QUEL

- arithmetr. Ausdrücke (wie in SQL) in RETRIEVE und in Bedingung erlaubt
- Was kostet durchschnittlich der Artikel 0815?
  RETRIEVE (Durchschnitt = AVG(L.Preis
  WHERE L.Artnr = 0815))
- Gruppierung mittels BY: Wieviele Lieferanten gibt es pro Artikel?
  RETRIEVE (L. Artnr, Anzahl = COUNT (L.LName
  BY L.Artnr))

6.5.5 QBE

- Query-By-Example, IBM-eigene Konkurrenz zu SQL
- ist ebenfalls IQL + DDL + DML + VDL + SSL, hier nur IQL
- Idee: Anfrage formulieren durch Ausfüllen einer Tabelle mit Variablen und Konstanten, basiert auf Domänenrelationenkalkül
- Variablen sind $\exists$-quantifiziert, Name mit _ eingeleitet, stellen beispielhafte Werte aus der Domäne dar
- Konstanten: ohne _
- Konstante oder Variable mit Präfix `P.` werden ausgegeben
- Variablen, die nur einmal in Anfrage auftauchen, können durch blanks ersetzt werden

Beispielanfragen:

- SELECT Artname FROM Artikel; entspricht:

Artikel

Artnr	EAN	Artname
		P._otto

- SELECT EAN FROM Artikel WHERE Artnr = 4711; entspricht:

Artikel

Artnr	EAN	Artname
4711	P._ean	

- Verbund durch

Artikel

Artnr	EAN	Artname
_999		

liefert

LName	Artnr	Preis
	_999	

QBE

- Welche Artikel(Nr.) werden von Lieferanten aus Osnabrück oder Münster geliefert?

liefert

LName	Artnr	Preis
_meyer	P.	
_mueller	P.	

Lieferant

LName	LAdresse
_meyer	Osnabrueck
_mueller	Muenster

- Welche Artikel(Nr.) werden von Lieferanten aus Osnabrück und Münster geliefert?

liefert

LName	Artnr	Preis
_meyer	P._999	
_mueller	P._999	

Lieferant

LName	LAdresse
_meyer	Osnabrueck
_mueller	Muenster

- Gib alle Artikel (Nr.) aus, deren Artikelname 'Hemd' enthält

Artikel

Artnr	EAN	Artname
P.		_xy Hemd_xz

- Gib Namen und Preise der Artikel aus, die Meyer liefert

Artikel	Artnr	EAN	Artname
	_999		_nam

liefert	LName	Artnr	Preis
	'Meyer'	_999	_pr

Ergebnis	Name	Preis
P.	_nam	_pr

QBE

- kompliziertere Bedingungen: mittels Condition-Box
  Welche Artikelnr. haben die gleiche Bezeichnung (Artname)?

Artikel	Artnr	EAN	Artname
	_999		_nam
	_998		_nam

Conditions
_999 < _998

Ergebnis	Artnr1	Artnr2
P.	_999	_998

- Sortiere die liefert-Relation aufsteigend nach LName und dann aufsteigend nach Artnr

liefert	LName	Artnr	Preis
	AO1	AO2	

  AO: Ascending Order, DO: Descending Order

- Was ist der durchschnittliche Preis vom Artikel Nr. 4711?

liefert	LName	Artnr	Preis
		4711	P.AVG.ALL._pr

- Was ist der durchschnittliche Preis pro Artikelnr.?

liefert	LName	Artnr	Preis
		P.G.	P.AVG.ALL.

6.5.6 Weitere Anfragesprachen

1. Universalrelationen-Anfragesprachen:

   - Idee: Formulierung der Anfragen ohne explizite Verbunde
   - vorzustellen: Universalrelation mit allen in DB vorhandenen Attributen
   - noch nötige Operationen: i.w. Projektion, Selektion
   - Bsp.: DB-Schema mit Relationen Studenten, Prüfer, Prüfungen

     statt

     ```
     SELECT SName
     FROM Studenten, Pruefer, Pruefungen
     WHERE Studenten.Matrnr = Pruefungen.Matrnr
     AND Pruefungen.Pruefernr = Pruefer.Pruefernr
     AND Pruefer.PName = 'Schreiner';
     ```

     nur kurz

     ```
     SELECT SName
     WHERE PName = 'Schreiner';
     ```

   - zugehörige Relationen und Verbunde muß System selber bestimmen
   - Voraussetzungen: Attributnamen global eindeutig, Schema azyklisch

Weitere Anfragesprachen

2. Regelbasierte Anfragesprachen:

- Idee: wie regelbasierte Programmiersprachen (z.B. PROLOG)
- Grundlagen (hier von DATALOG):
  - Prädikate (Atome): beschreiben Relation oder eine Selektion auf einer Relation (abgeleitete Relation); daneben: Vergleichsprädikate der Form X = Y, X < Y, X < 3, ...
  - Literal: Atom oder negiertes Atom
  - Klausel: Disjunktion von Literalen
  - Horn-Klausel: Klausel mit höchstens einem nicht-negativen Literal;
    * nur ein nicht-negatives Literal: Fakt
    * kein nicht-negatives Literal: Integritätsbedingung
    * ein nicht-negatives und ein oder mehrere negative Literale: Regel
    * Beachte: $\neg P_1 \vee ... \vee \neg P_n \vee P \iff$
      $P_1 \wedge ... \wedge P_n \Rightarrow P$
      geschrieben als:
      P :- P_1, ..., P_n.
- DATALOG-Programm: Menge von Horn-Klauseln
- Variablen die nur im Rumpf vorkommen sind $\exists$-quantifiziert, alle anderen $\forall$-quantifiziert