KfK 3089
März 1981

Die Datenbankanfragesprache FAQUEL

F. J. Polster, P. Tritschler
Institut für Datenverarbeitung in der Technik

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Datenverarbeitung in der Technik

KfK 3089

Die Datenbankanfragesprache

FAQUEL

Franz J. Polster
Peter Tritschler

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
ISSN 0303-4003

Zusammenfassung

FAQUEL ist die interaktive, selbständige Anfragesprache des Datenbanksystems FADABS. Das FAQUEL-Anfragesystem dient als Komponente von Prozeßinformationssystemen: Es soll der Benutzerklasse der "anspruchsvollen EDV-Laien" - Physiker, Chemiker, technisches Personal - ermöglichen, lesend und ändernd ohne vorherige Programmierung auf FADABS-Datenbanken zuzugreifen.

Ziel des Entwurfs war es, eine Anfragesprache zu entwickeln, die einfach erlernbar und verständlich ist und sich für "echte" Anwendungssysteme eignet. Als Vorbilder dienten Sprachen wie TAMALAN und SEQUEL2.

Die folgenden Konzepte sind besonders wichtig für die Anwendung in der Praxis:

- Zwischenergebnisse können in Zwischenrelationen abgelegt werden: dies erlaubt zum einen die Zerlegung komplexer Anfragen in mehrere, einfachere Schritte, für die dann entsprechend einfache FAQUEL-Anweisungen genügen; zum andern ermöglicht die Benutzung bereits vorliegender Zwischenergebnisse ein effizienteres Arbeiten
- Der Tupelmodus erlaubt ein Navigieren in der Datenbank, ein weiteres Mittel zur Beantwortung schwer oder umständlich formulierbarer Anfragen in mehreren einfachen Schritten. Vor allem aber wird das Arbeiten mit "langen" Tupeln erleichtert und vereinfacht, ein für die Praxis wichtiger Aspekt.
- FAQUEL-Anweisungen zu häufig wiederkehrenden Anfragen können zu Prozeduren zusammengefaßt und einfach durch Angabe des Prozedurnamens ausgeführt werden
- Gruppierte Relationen erlauben mit einfachen Anweisungen die Formulierung von Anfragen, die der Division der Relationenalgebra entsprechen.

The Database Query Language FAQUEL

Abstract

FAQUEL is the interactive, selfcontained query language of the database system FADABS; the FADABS query system is a component of process information systems. It is to provide the user group of "nonprogrammers" - physicists, chemists, technicians - with convenient features for the retrieval of data from and manipulation of FADABS databases in an interactive fashion, i.e. without prior programming.

The main goal was the development of a query language which is easy to learn and understand and applicable in typical real world situations. The design was influenced by languages like TAMALAN and SEQUEL2.

The following concepts are useful especially for the practitioner:

- Intermediate results can be saved in intermediate relations for subsequent processing. This feature allows to express semantically complicated questions as a series of several simpler queries, therefore the FAQUEL statements can be simple. It may also increase the efficiency of the FAQUEL-user, since data retrieved and used in the course of preceeding queries can be reused immediately.
- The tuplemode enables the user to navigate a database, a means for expressing complicated problems or otherwise clumsy or lengthy FAQUEL-statements by several simpler statements. Also it provides features for the convenient handling of large tuples typical for real-world applications
- A set of FAQUEL statements can be defined a procedure and executed by simply entering the procedure name, a means of reducing the amount of typing necessary for frequently recurring queries
- Grouped relations provide for the formulation of queries equivalent in power to the Relational Algebra's Division by a set of simple FAQUEL-statements.

Inhalt:

Einleitung

FADABS ist ein allgemeines, anwendungsunabhängiges Datenbanksystem, das am Institut für Datenverarbeitung in der Technik für den Einsatz auf Kleinrechnern entwickelt worden ist /Po 78/.

Es wird u.a. als Komponente von Prozeßinformationssystemen eingesetzt; wie in /JPT 80/ erläutert unterstützt FADABS vier verschiedene Anwenderschnittstellen:

a) Die DBA-Schnittstelle für den Datenbankadministrator.

   Sie stellt die Datendefinitionssprache (u.a. Einrichten/Löschen von Relationen) und administrative Funktionen zur Verfügung /Ke 79/.

b) Die data sublanguage FORDAM für den FORTRAN Anwendungsprogrammierer /Po 80/.

   Hiermit können spezifische, an die jeweiligen Erfordernisse angepaßte Datenbankschnittstellen unter Benutzung von FORTRAN programmiert werden, z.B. Dialoge für "parametrische Benutzer" /We 74/, Programme zur online Meßdatenerfassung oder Datenauswertungen.

c) Die Report-Definitionsprache RDL des Report-Generators FAREG zur Beschreibung und Erzeugung von Listen (reports) über Daten aus FADABS Datenbanken /SP 80/.

d) Die Verwendung von FORDAM impliziert ganz offensichtlich Programmerstellung, also Compilieren, Binden, Laden, dies gilt ebenfalls - wenn auch in geringerem Umfang - für die Report-Erstellung mit FAREG. In Prozeßinformationssystemen muß es i.a. aber auch möglich sein, Daten (z.B. Meßwerte von Versuchsreihen) nach beliebigen, a priori nicht vorhersehbaren, und damit nicht vorprogrammierbaren Kriterien und Strategien zu durchsuchen, ohne hierzu erst neue Programme entwickeln zu müssen.

   Zu diesem Zweck wurde das interaktive Datenbankanfragesystem FAQUEL mit der gleichnamigen Anfragesprache geschaffen (FAQUEL: FADABS query language).

In diesem Bericht wird die Anfragesprache FAQUEL beschrieben. Abschnitt 1 geht auf die prinzipiellen Überlegungen und Kriterien beim Entwurf von FAQUEL ein, die eigentliche Sprachbeschreibung erfolgt in den Abschnitten 2 bis 8.

In Abschnitt 9 wird der Bezug zum relationalen Datenmodell von Codd /Co 70/ hergestellt und die Frage der relationalen Vollständigkeit behandelt. Die Implementierung von FAQUEL, also das Anfragesystem, wird in Abschnitt 10 kurz skizziert.

Diese Arbeit basiert auf der Diplomarbeit des zweiten Autors /Tr 79/.

1. FAQUEL: Ziel, Entwurfskriterien

1.1. Die FAQUEL-Benutzer

Hauptziel der Entwicklung der Anfragesprache FAQUEL war es, den Anwendern von Prozeßinformationssystemen FADABS-Datenbanken unmittelbar, d.h. ohne vorherige Programmierung, zugänglich zu machen. Diese Anwender sind Physiker, Chemiker, Ingenieure, Techniker, Inspektoren von EURATOM und IAEA, von denen keine intensiven EDV-Kenntnisse vorausgesetzt werden können. Auf Grund ihrer naturwissenschaftlichen Ausbildung sind sie mit formalen Darstellungsweisen vertraut. Einer Einteilung der Benutzer von Datenbanksystemen in 5 Klassen zufolge, wie sie bei WEDEKIND /We 74/ zu finden ist, gehören sie zur Benutzerklasse der "anspruchsvollen Laien".

Damit ist natürlich nicht ausgeschlossen, daß auch EDV-Spezialisten, etwa als Testhilfsmittel während der Erstellung eines Prozeßinformationssystems, von FAQUEL Gebrauch machen werden. Das Hauptaugenmerk war jedoch darauf gerichtet, ein den Anforderungen der eigentlichen Anwender entsprechendes Anfragesystem zu entwerfen.

1.2. Der Anforderungskatalog

Die wesentlichsten Punkte des Anforderungskatalogs waren (vergl. /Tr 79/):

1.2.1. Allgemeine Forderungen an die Anfragesprache

a) FAQUEL soll einfach erlernbar, verwendbar und leicht verständlich sein. Das System soll interaktiv, als Dialogsystem arbeiten.

b) Aus Gründen der Portabilität /Po 78/ und Vielseitigkeit sollen von den Dialoggeräten keine besonderen Hardware-Eigenschaften gefordert werden: neben Sichtgeräten sollen z.B. auch teletype-Terminals als Dialoggeräte eingesetzt werden können.

c) Die Ergebnisse sollen - dies ist bedeutsam vor allem beim Bildschirmbetrieb - auch über einen Drucker auf Papier ausgegeben werden können; ebenso soll auf Wunsch der gesamte Dialogablauf einer Sitzung auf dem Drucker mitprotokolliert werden.

1.2.2. Fähigkeiten des Anfragesystems

a) FAQUEL wird vorwiegend für Retrievaloperationen benutzt werden; folgende Möglichkeiten sind bereitzustellen:

   - Auswahl der am Dialoggerät auszugebenden Attributwerte eines Tupels (d.h. es werden i.a. nicht alle Attribute der gewünschten Tupel benötigt)
   - Auswahl der gewünschten Tupel mittels Qualifikationen unter Verwendung der Vergleichsoperatoren: $= \neq < \leqq \geqq >$ und Booleschen Operatoren: AND, OR. Dabei werden sowohl Attribut-Konstante- wie auch Attribut-Attribut- Vergleiche gewünscht.
   - Verknüpfung von Tupeln zweier Relationen
   - "Navigieren" in der Datenbank (vergl. auch Abschnitt 5.1.)
   - Sortierung
   - Auswertung von Daten durch arithmetische Ausdrücke (arith. Grundoperationen) und Aggregatfunktionen wie Summe, Durchschnitt.

b) Anweisungen zum Einfügen neuer Tupel, Ändern oder Löschen existierender Tupel

c) Definition und Ausführung von FAQUEL-Prozeduren.

Dabei ist zu beachten, daß die Relationen der bereits implementierten oder konzipierten Anwendungssysteme aus einer großen Anzahl von Attributen (zum Teil über 40!) bestehen. Weiter ist zu bedenken, daß FAQUEL auf Kleinrechnern laufen soll (Einleitung, 1. Absatz) und, um die Akzeptanz zu sichern, die Effizienz des Systems nicht ganz ohne Beachtung bleiben darf.

1.2.3. Nicht gefordert waren:

- Möglichkeiten zur Datendefinition, da dies ausschließlich dem DBA überlassen bleiben soll /Po 78/.
- spezielle Anweisungen zur Formatierung der Ausgabe oder Listenerzeugung. Hierfür wurden der Report-Generator FAREG /SP 80/ geschaffen.

1.3. Allgemeine Überlegungen zum Sprachentwurf

1.3.1. Die Anforderungen von 1.2.2. werden durch Bereitstellung entsprechender FAQUEL-Anweisungen oder Kommandos erfüllt, diese werden in den nachfolgenden Abschnitten vorgestellt. Es soll hier kurz darauf eingegangen werden, wie durch den Entwurf dieser Anfragesprache versucht worden ist, den Anforderungen vom 1.2.1. gerecht zu werden.

Um die Forderungen von 1.2.1. zu erfüllen, sind außer der genauen Kenntnis der Benutzerklasse, grundlegende Kenntnisse über den Prozeß der interaktiven Anfrageformulierung an eine Datenbasis nötig. Die Untersuchungen hierzu befinden sich noch in den Ansätzen, in /Re 77, Lo 77, Sh 78, Th 75, Mc 75/ hat man unter anderem versucht, diesen Prozeß zu ergründen, Fehlerarten und -ursachen zu analysieren, Sprachmerkmale herauszufinden, die das Verständnis und die Erlernbarkeit einer Sprache erschweren.

Beim Entwurf von FAQUEL wurde versucht, folgende Erkenntnisse aus diesen Arbeiten zu berücksichtigen:

a) Je mehr die Benutzerklasse in Richtung "gelegentliche Benutzer" /We 74/ tendiert, desto geeigneter sind deskriptive Sprachen gegenüber prozeduralen /Lo 77/.

b) Die Verwendung von Quantoren und Variablen bereitet vielen Benutzern Schwierigkeiten /Re 77/.

c) Die Verwendung von Grundformaten für Anfragen unterstützt die Erlernbarkeit und das Behalten der Sprachkonstruktionen /Lo 77/.

d) Die Verwendung von Schlüsselwörtern unterstützt das Erlernen und Formulieren von Anfragen, indem sich die Semantik der Anfragen in den Schlüsselwörtern wiederfindet. Schlüsselwörter werden gegenüber anderen Delimitern (wie Komma, Semikolon, Apostroph, u.ä.) vor allem von Nichtprogrammierern und Benutzern mit weniger mathematischer Vorbildung bevorzugt /Lo 77/.

e) Semantisch unterschiedliche Konzepte sollen durch syntaktisch unterschiedliche realisiert werden /Lo 77/.

f) Für einfache Anfragen sollten einfache Sprachelemente ausreichen, komplexere Anfragen sollen durch eventuell entsprechend kompliziertere und schwerer verständliche Sprachkonstruktionen formulierbar sein. Für den ungeübten Benutzer genügt es dann, zunächst die einfacheren Sprachelemente zu erlernen /Re 77/.

1.3.2. Für den Entwurf von FAQUEL ergaben sich hieraus folgende prinzipiellen Festlegungen: FAQUEL ist eine im wesentlichen deskriptive, abbildungsorientierte Anfragesprache, bei der die Sprachfunktionen durch englische Schlüsselwörter ausgedrückt werden (Zur Definition von "deskriptiv", "abbildungsorientiert" vergl. man /We 74/).

Eine abbildungsorientierte Sprache scheint am besten geeignet, der Benutzerklasse der "anspruchsvollen Laien" eine einfach zu verwendende, leicht verständliche und leicht erlernbare Sprache zu sein /Lo 77/. Sie erfordert weniger mathematisches Verständnis als eine prozedurale, relationenalgebraorientierte oder eine relationenkalkülorientierte Sprache und kommt ohne Quantoren aus. Eine graphikorientierte Sprache wie QbE /Z1 75/ kommt nicht in Frage, da die Relationen der FADABS-Anwendungsysteme aus sehr vielen Attributen bestehen (vergl. Feststellung von 1.2.2.) und sich nicht mehr in geeigneter Weise auf dem Bildschirm darstellen lassen, wie für QbE erforderlich. Andere graphikorientierte Sprachen, wie z.B. CUPID, erfordern spezielle Hardware (Bildschirm mit Lichtgriffel), die nicht vorausgesetzt werden sollte.

In FAQUEL wird ein Grundformat für Retrievalanweisungen und eines zur Definition von Zwischenrelationen verwendet. Durch Ergänzung der Grundformen gelangt man von einfachen zu komplexeren Anfragen.

Syntax und Semantik von FAQUEL orientieren sich an den beiden abbildungsorientierten Sprachen TAMALAN /Va 77/ und SEQUEL2 /Ch 76/.

FAQUEL kennt folgende Konzepte:

Zwischenrelationen:

Von TAMALAN wurde das Konzept der Zwischenrelation übernommen. Zwischenrelationen sind temporäre Relationen (temporär für die Dauer einer Dialogsitzung oder bis zur expliziten Löschung durch eine Anweisung), die sich der Benutzer definieren kann. Sie geben ihm die Möglichkeit, komplexe Anfragen in Einzelschritte aufzuspalten und Zwischenergebnisse in solchen temporären Relationen abzulegen. Dadurch lassen sich kompliziert geschachtelte Anfrageformulierungen vermeiden. Durch Verwendung von Zwischenrelationen läßt sich die Einführung von Variablen vermeiden, wie sie z.B. in SEQUEL in bestimmten Fällen erforderlich sind (Abschnitt 6).

Gruppierung:

Von SEQUEL2 wurde das Gruppierungskonzept und die Mengenfunktion übernommen, mit deren Hilfe sich die relationenalgebraische Division realisieren läßt (Abschnitt 7 und 9).

Tupelmodus:

Es wird ein tupelweises Arbeiten ermöglicht. Dabei geht nach Ausgabe jeweils eines Tupels die Kontrolle an den Benutzer zurück und er kann noch vor der vollständigen Ausgabe des Ergebnisses eine neue Retrieval- oder Manipulationsanweisungen eingeben (Abschnitt 5).

Prozeduren:

Eine oder mehrere Anweisungen lassen sich als Prozedur zusammenfassen. Zur Ausführung genügt der Aufruf mit einem Prozedurnamen (Abschnitt 8.5).

2. Die Elemente von FAQUEL

2.1. Das Datenmodell

FAQUEL greift auf FADABS Datenbanken zu; die wesentlichen Begriffe sind Relation, Tupel, Attribut:

Eine FADABS Relation ist eine n-stellige Relation $R = R(A_1, A_2, \ldots, A_n)$. Die A_i sind die Attributnamen oder auch kurz die Attribute und bezeichnen Mengen von (Attribut-)Werten. Die Elemente t einer Relation R heißen Tupel, der Wert der i-ten Komponente eines Tupels t aus R ist ein Element der Menge A_i. Die Attributwerte eines Attributs sind von einem bestimmten Typ, dem "Attributtyp"; in Tabelle 1 sind die möglichen FADABS-Attributtypen aufgeführt und unter Verwendung von FORTRAN-Begriffen erläutert.

Es gibt einfache und komplexe Attribute: komplexe Attribute sind stets vom Typ A*n und stellen eine Zusammenfassung einfacher Attribute dar (vergl. /Ke 79/). Hinweis: Informationen über die Attribute einer Relation erhält der Benutzer durch Anfordern der Relationenbeschreibung (Abschnitt 8.3).

Attributtyp	Wertebereich ("domain")
I*2	INTEGER*2-Zahlen
I*4	INTEGER-Zahlen
R*4	REAL-Zahlen
R*8	DOUBLE PRECISION-Zahlen
A*n (n > 0)	Zeichenketten der Länge n, begrenzt durch ' (Leerzeichen zwischen zwei ' sind signifikant!)

Tabelle 1: FADABS-Attributtypen

Relationen werden häufig als Tabellen dargestellt: Eine Zeile der Tabelle entspricht dabei einem Tupel, eine Spalte enthält die Werte zu einem Attribut.

Die Möglichkeiten von FAQUEL werden im folgenden anhand von Beispielen veranschaulicht. Es wird hierzu eine fiktive Anwendung aus dem kerntechnischen Bereich angenommen, bei der Kernmaterial in Form von "Plättchen" in einem Lager zu verwalten ist. Die Datenbank besteht aus drei Relationen:

Die Relation KEMAT mit vier Attributen zur Beschreibung der möglichen Plättchenarten:

- MATNAM, ein A*10 Attribut, gibt den Materialnamen, hier also den Namen eines Plättchentyps, an (Zeichenkette der Länge 10)
- Die R*4 Attribute UGES, U235, PUGES bezeichnen in Gramm die gesamte Masse Uran bzw. die Masse des U-Isotops 235 bzw. die gesamte Masse Plutonium des Plättchens.
- VERTRG, ein A*2 Attribut, für den Vertrag, unter dem die Plättchen eingeführt worden sind.

Die Relation LAGER mit drei Attributen beschreibt, welche und wieviele Plättchen sich in einer Lagerposition befinden:

- ORT, ein A*10 Attribut, gibt die Bezeichnung einer Lagerposition an
- MATNAM, ein A*10 Attribut, gibt wie bei KEMAT den Plättchennamen an
- ANZAHL, ein I*2 Attribut, liefert die Zahl von Plättchen an der betreffenden Position.

Die Relation LAGNAM mit dem einzigen Attribut ORT vom Typ A*10 wird als Hilfsrelation zum Zwischenspeichern von Lagerpositionsbezeichnungen benutzt (Abschnitt 4.1.5).

Die Beispiele benutzen KEMAT von Bild 1 und LAGER von Bild 2: so entnimmt man z.B. Bild 1, daß ein Plättchen PU NP6 87.85g (0.62g, 31.81g) Uran (U-Isotop 235, Plutonium) enthält und nach Bild 2 23 bzw. 18 Plättchen dieser Art in LA/1/1/1 und LA/2/1/9 lagern.

MATNAM	UGES	U235	PUGES	VERTRG
PU NP6	87.85	0.62	31.81	V1
PU VAK NP3	0.0	0.0	55.74	V2
U-35% KP3	124.70	43.64	0.0	V3
U-NAT NB25	1232.00	8.76	0.0	V4
UO2 NP6	127.00	0.90	0.0	V5
PU NPX	103.49	36.12	64.79	V6

Bild 1: Die Relation KEMAT

ORT	MATNAM	ANZAHL
LA/1/1/1	PU NP6	23
LA/1/1/1	U-35% KP3	5
LA/1/1/1	PU NPX	31
LA/2/1/9	PU NP6	18

Bild 2: Die Relation LAGER

2.2. FAQUEL-Sprachelemente

Die Elemente von FAQUEL-Anweisungen sind Bezeichner, Konstanten, Schlüsselwörter und Sonderzeichen.

2.2.1. Bezeichner, Schlüsselwörter

Ein FAQUEL-Bezeichner besteht aus bis zu 6 Buchstaben und/oder Ziffern, wobei das erste Zeichen ein Buchstabe sein muß.

Bezeichner sind symbolische Namen für

- Relationen, Zwischenrelationen
- Attribute, Attributlisten
- Prozeduren, Parameter

Bezeichner dürfen nicht mit dem relevanten Teil eines Schlüsselwortes (die ersten 6 Zeichen) identisch sein.

Schlüsselwörter sind fest vereinbarte Zeichenfolgen, denen eine bestimmte Bedeutung zukommt. Relevant sind nur die ersten 6 Zeichen!

Eine Liste aller Schlüsselwörter von FAQUEL ist im Anhang 2 enthalten.

2.2.2. Konstanten

FAQUEL kennt numerische und string-Konstanten: Numerische Konstanten sind I*2-,I*4-,R*8- Konstanten. Sie entsprechen in Schreibweise und Wertebereich den FORTRAN-Konstanten vom Typ INTEGER*2, INTEGER, REAL, DOUBLE PRECISION (REAL*8) /ANS 77/.

String-Konstanten, im folgenden auch als "Zeichenketten" bezeichnet, entsprechen den CHARACTER-Konstanten von FORTRAN /ANS 77/ (Zeichenfolge eingeschlossen in ').

2.2.3. Sonderzeichen

Alle in FAQUEL vorkommenden Sonderzeichen sind in Anhang 2 aufgeführt.

2.2.4. Arithmetische Ausdrücke

Ein arithmetischer Ausdruck besteht aus den Operatoren +, -, *, / und einem oder mehreren Operanden. Die Operatoren haben die übliche mathematische Bedeutung. Operanden sind Konstanten und Attributwerte. In Verbindung mit den Operatoren sind nur numerische Operanden zulässig; ein arithmetischer Ausdruck kann aber auch aus einem A*n-Attribut als einzigem Operanden bestehen. Die Syntax der arithmetischen Ausdrücke ist wie folgt definiert (die Notation ist in 2.3. erläutert):

```
arith-expr      ::= {-} term |{-} term aop arith-expr
term            ::= factor | factor mop term
factor          ::= num-const | attr-name | (arith-expr)
aop             ::= + | -
mop             ::= * | /
num-const       ::= int-const | real-const | double-const
```

int-const, real-const, double-const sind Konstanten vom Typ I*2, I*4, R*4, R*8 (vergl. 2.2.2.).

Jede Operation des arithmetischen Ausdrucks liefert ein Ergebnis von einem bestimmten Typ. Dieser Ergebnistyp ist gleich dem Typ der beteiligten Operanden. Bei unterschiedlichen Operandentypen wird ein Operand in den Typ des anderen Operanden umgewandelt gemäß Tabelle 2 und danach die Operation ausgeführt.

1. Operand / 2. Operand	I*2	I*4	R*4	R*8
I*2	I*2	I*4	R*4	R*8
I*4	I*4	I*4	R*4	R*8
R*4	R*4	R*4	R*4	R*8
R*8	R*8	R*8	R*8	R*8

Tabelle 2: Typkonversionen bei arithmetischen Operationen

2.2.5. Funktionen

Ein Funktionsaufruf besteht aus einem Funktionsbezeichner func-name und einem Parameter attr-name, stets ein Attributname.

Syntax:

```
func          ::= func-name (attr-name)
func-name     ::= MAX | MIN | SUM | AVG | COUNT
```

Die Bedeutung, Werte- und Bildbereiche der einzelnen Funktionen sind Tabelle 3 zu entnehmen. Weitere Erläuterungen zu den Funktionen finden sich bei den Anweisungen, in denen Funktionen verwendet werden können.

Funktion	Bedeutung	Parametertyp	Typ des Funktionswertes
MAX	Maximum	I*2	I*2
MIN	Minimum	I*4	I*4
		R*4	R*4
		R*8	R*8
		A*n	A*n
SUM	Summe	I*2	I*2
		I*4	I*4
		R*4	R*4
		R*8	R*8
AVG	Durch-	I*2	R*4
	schnitt	I*4	R*4
		R*4	R*4
		R*8	R*8
COUNT	Anzahl	I*2	I*2
		I*4	I*2
		R*4	I*2
		R*8	I*2
		A*n	I*2

Maximum (Minimum) von Zeichenketten: Das Maximum (Minimum) einer Menge von Zeichenketten, ist die Zeichenkette, die gemäß der lexikographischen Ordnung an letzter (erster) Stelle steht (vergl. 2.2.7).

Tabelle 3: FAQUEL-Funktionen

2.2.6. Typverträglichkeit

In FAQUEL ist ein Operand ein Attributwert, eine Konstante, ein Funktionswert oder das Ergebnis eines arithmetischen Ausdrucks. Jeder Operand ist von einem bestimmten Typ. Bei Vergleichsoperationen (2.2.7) der Art

Operand 1 rop Operand 2 rop: Vergleichsoperator

müssen die Typen der Operanden miteinander verträglich sein.

Ebenso wird beim Einfügen von Tupeln in eine Relation (Abschnitt 4.1.1, 5.2.5.) gefordert, daß der Typ des i-ten Attributs (Operand 1) mit dem Typ des i-ten Attributwertes (Operand 2) des neuen Tupels verträglich ist.

Die Typverträglichkeit wird durch Tabelle 4 definiert.

Typ Operand 2 \ Typ Operand 1	I*2	I*4	R*4	R*8	A*n
I*2	V	V	V	V	NV
I*4	NV	V	V	V	NV
R*4	NV	NV	V	V	NV
R*8	NV	NV	NV	V	NV
string-Konstante	NV	NV	NV	NV	V

V = verträglich
NV = nicht verträglich

Tabelle 4: Typverträglichkeit

Bemerkungen:

Sind der Typ von Operand 1 und Operand 2 ungleich aber verträglich, so wird Operand 2 konvertiert in den Typ von Operand 1 (Typausweitung) und danach die Operation ausgeführt.

Bei Operanden, deren Wert ein string ist, darf Operand 2 höchstens so lang sein wie Operand 1. Ist Operand 2 länger, so wird dies als Fehler gemeldet, ist er kürzer, so wird er bis auf die Länge von Operand 1 als nach rechts mit Leerzeichen ergänzt betrachtet.

2.2.7. Vergleichsoperationen

Vergleichsoperationen haben die Form

Operand 1 rop Operand 2

rop: Vergleichsoperator EQ (=), NE (≠), LT (<), LE (<=)
GT (>), GE (>=)

Die beiden Operandentypen müssen dabei mindestens verträglich (2.2.6.) sein; in bestimmten Fällen wird sogar Gleichheit gefordert (siehe z.B. 3.1.3.). Bei numerischen Operanden (Typ I*2, I*4, R*4, R*8) gilt die übliche kleiner/gleich -Relation zwischen Zahlen. Beim Vergleich von Zeichenketten wird von der üblichen lexikographischen Ordnung ausgegangen. Dabei darf Operand 2 höchstens so lang sein wie Operand 1. Ist Operand 2 länger, so wird dies als Fehler gemeldet, ist er kürzer und wird keine Typgleichheit gefordert, so wird er bis auf die Länge von Operand 1 als nach rechts mit Leerzeichen ergänzt betrachtet.

2.3. Zur Sprachbeschreibung

In den folgenden Abschnitten werden Syntax und Semantik der Sprachelemente von FAQUEL beschrieben und anhand von Beispielen verdeutlicht. Die Beispiele beziehen sich auf die Beispieldatenbasis von 2.1.

Die Syntax der FAQUEL-Anweisungen wird zunächst nur soweit angegeben, wie es zum Verständnis der jeweils vorgestellten Sprachkonstruktion notwendig ist. Dabei wird von einer BNF-ähnlichen Notation Gebrauch gemacht:

- ::= und | haben die übliche Bedeutung /Gr 71, AU 72/
- { string } $\begin{smallmatrix} n \\ m \end{smallmatrix}$ bedeutet, daß string mindestens m-mal, höchstens n-mal auftritt.
  { string } ist gleichbedeutend mit { string } $\begin{smallmatrix} 1 \\ o \end{smallmatrix}$
- Nichtterminalsymbole sind in Kleinbuchstaben geschrieben
- Schlüsselwörter sind in Großbuchstaben geschrieben

Eine Beschreibung der FAQUEL-Syntax in Form von Syntaxdiagrammen enthält Anhang 1.

3. Retrieval-Anweisungen

Zur Formulierung von Datenbankanfragen stehen LIST-Anweisungen zur Verfügung, sie erlauben das Lesen und Auswerten von Daten einer Datenbank.

Jede LIST-Anweisung beginnt mit dem Schlüsselwort "LIST", gefolgt von der Beschreibung der Daten, nach denen gefragt wird, der "Ergebnisbeschreibung".

Die LIST-Anweisung ist so entworfen, daß sich die Kompliziertheit einer Anfrage in der Komplexität der entsprechenden Ergebnisbeschreibung widerspiegelt: Einfache Fragen lassen sich durch einfache, i. a. kurze LIST-Anweisungen ausdrücken, schwierigere Fragen erfordern die Benutzung komplexerer Sprachmittel und führen i. a. zu längeren, aufwendigeren LIST-Anweisungen (vergl. hierzu 1.3.).

Im folgenden werden ausgehend von den einfachsten Formen der LIST-Anweisung die verschiedenen Möglichkeiten zur Formulierung von Datenbankanfragen dargestellt.

Erläuterung zur Ausgabe der Ergebnisdaten:

Mit LIST-Anweisungen wird eine "Ergebnisrelation" spezifiziert, die nur auf dem Dialoggerät ausgegeben werden kann und anschließend nicht mehr existiert; insbesondere also hat sie keinen Namen. Die Ausgabe der Ergebnisrelation erfolgt in Blöcken:

- Jeder Block besteht aus n Tupeln dieser Relation, n kann mit der Anweisung

  TUPLE n;

  gesetzt werden, n < 32768.

  Nach Ausgabe eines Blocks kann der Benutzer
  - den nächsten Block anfordern: Drücken der Datenübertragungstaste (DÜZ-Taste bei SIEMENS Anlagen)
  - auf weitere Ausgabe verzichten: Eingabe von CANCEL;
- Jedes Tupel wird in 1 Zeile des Ausgabegeräts ausgegeben, die Tupellänge (Zahl der auszugebenden Zeichen einschließlich Leerzeichen für Zwischenräume) darf daher nicht größer als die Zeilenlänge des Ausgabegeräts sein.

- Jeder Block erhält eine Blocküberschrift:
  Neben dem Relationennamen wird zu jeder Spalte der Attributname des Attributs angegeben, dessen Werte die Spalte enthält.
  Hinweis: Mit Attributumbenennungen kann eine Spaltenüberschrift explizit definiert werden (vergl. Hinweis von 3.1.2.).

3.1. Einfache LIST-Anweisungen

3.1.1. Vollständiges Auflisten einer Relation

Syntax:

```
LIST rel-name;
```

Semantik:

Es werden sämtliche Tupel der Relation rel-name mit allen Attributwerten ausgegeben.

Beispiel:

```
LIST KEMAT;
```

Ergebnis: Diese Anweisung bewirkt das vollständige Auflisten auf dem Dialoggerät aller Tupel der Relation KEMAT, wie in Bild 1 angegeben.

3.1.2. Auswahl von Spalten einer Relation

Syntax:

```
LIST attr-list  OF  rel-name;
attr-list ::= attr |  attr , attr-list
attr      ::=      { name   = } attr-name
```

Semantik:

Es werden von sämtlichen Tupeln der Relation rel-name nur die Attributwerte der mit attr-list spezifizierten Attribute attr-name ausgegeben: attr-list

ist eine Liste, die aus Attributnamen und/oder Attributumbenennungen name = attr-name besteht; jedes attr-name muß ein Attribut der Relation rel-name sein. In der Blocküberschrift wird für jedes Attribut einer Attributumbenennung name (und nicht mehr attr-name!) aufgeführt.

Hinweis: Mit Attributumbenennung kann man also in gewissem Rahmen die Blocküberschrift explizit definieren.

Beispiele:

Anfrage: Welche Plättchen gibt es und wieviel U235 enthält jedes einzelne?

a) LIST MATNAM, U235 OF KEMAT;

Ergebnis:

MATNAM	U235
PU NP6	0.62
PU VAK NP3	0.0
U-35% KP3	43.64
U-NAT NB25	8.76
UO2 NP6	0.90
PU NPX	36.12

b) LIST MATNAM, URAN35 = U235 OF KEMAT;

Ergebnis: wie a), jedoch: in der Überschrift steht "URAN35" anstelle von "U235"

3.1.3. Auswahl von Tupeln einer Relation

Syntax:

```
LIST rel-name WHERE t-qual;
     t-qual          ::= t-qual-term  |  t-qual-term OR t-qual
     t-qual-term     ::= t-qual-factor  |  t-qual-factor AND t-qual-term
     t-qual-factor   ::= predicate | (t-qual)
     predicate       ::= attr-name rop const  |  attr-name-1 rop attr-name-2
     constant        ::= string-const  |  num-const
     rop             ::= LT | LE | EQ | GE | GT | NE
```

Semantik:

Es werden die vollständigen Tupel der Relation rel-name aufgelistet, die der (Tupel-)Qualifikation t-qual genügen.

Eine Qualifikation t-qual kann also aus einem oder mehreren Prädikaten (predicate) bestehen, die durch die Booleschen Operatoren AND, OR (in der üblichen Bedeutung) miteinander verknüpft sind; Klammerung, auch geschachtelt, ist erlaubt.

Die Vergleichsoperatoren rop haben (in Anlehnung an die FORTRAN-Notation) die folgende Bedeutung:

LT:	kleiner	LE:	kleiner/gleich
EQ:	gleich	NE:	ungleich
GT:	größer	GE:	größer/gleich

Für jedes Prädikat der Art attr-name rop const müssen der Typ des Attributs und der Konstanten miteinander verträglich sein; Typverträglichkeit ist durch Tabelle 4 in Abschnitt 2.2.6 beschrieben. Für jedes Prädikat attr-name-1 rop attr-name-2 müssen die Typen der beiden Attribute übereinstimmen.

Zur Erläuterung von Vergleichsoperationen siehe auch Abschnitt 2.2.7.

Ein Tupel aus rel-name erfüllt genau dann t-qual, wenn nach dem Ersetzen der Attributnamen in den Prädikaten durch die entsprechenden Attributwerte des Tupels der aus t-qual resultierende logische Ausdruck den Wert "wahr" ergibt.

Hinweis: In den Abschnitten 3.2.2, 3.3 wird diese vorläufige Definition von Tupelqualifikationen erweitert bzw. verallgemeinert, vergl. auch 3.7.

Beispiele:

Anfrage: Welche Plättchen enthalten kein Plutonium?

```
LIST KEMAT WHERE PUGES EQ 0.0;
```

Ergebnis:

MATNAM	UGES	U235	PUGES	VERTRG
U-35% KP3	124.70	43.64	0.0	V3
U-NAT NB25	1232.00	8.76	0.0	V4
UO2 NP6	127.00	0.90	0.0	V5

Anfrage: Welche Plättchen enthalten mehr Plutonium als U235?

```
LIST MATNAM OF KEMAT WHERE PUGES GT U235;
```

Ergebnis:

MATNAM
PU NP6
PU VAK NP3
PU NPX

Anfrage: Welche Plättchen enthalten entweder

- mehr als 500 g Uran oder
- mindestens 100 g Uran und mindestens 50 g Plutonium

```
LIST  KEMAT  WHERE  UGES  GT  500.0  OR
     (UGES  GE  100.0  AND  PUGES  GE  50);
```

Ergebnis:

MATNAM	UGES	U235	PUGES	VERTRG
U-NAT NB25	1232.00	8.76	0.0	V4
PU NPX	103.49	36.12	64.79	V6

3.1.4. Spalten- und Tupelauswahl

Es ist möglich, Spaltenauswahl (3.1.2.) und Tupelauswahl (3.1.3.) in einer LIST-Anweisung vorzunehmen.

Syntax:

```
LIST attr-list OF rel-name WHERE t-qual;
```

Semantik:

Es werden von den ausgewählten Tupeln nur die durch attr-list spezifizierten Attributwerte (d.h. Spalten) ausgegeben.

Beispiel:

Anfrage: Gesucht sind die Namen der Plättchen mit dem Anteil an U235 unter der Überschrift URAN35, die mehr als 500 g Uran oder mindestens 100 g Uran und mindestens 50 g Plutonium enthalten!

```
LIST  MATNAM, URAN35 = U235  OF  KEMAT
      WHERE  UGES  GT  500.0      OR
             (UGES GE  100.0     AND  PUGES  GE  50);
```

Ergebnis:

MATNAM	URAN35
U-NAT NB25	8.76
PU NPX	36.12

3.1.5. Ausgabe auf das Druckgerät

Der FAQUEL-Anwender wird in der Regel als Dialoggerät ein Sichtgerät benutzen. Aus folgenden Gründen kann er die Ausgabe der Ergebnisrelation auf dem Drucker wünschen:

- die Daten werden auf Papier, etwa zum Abheften (Archivieren) benötigt
- die auszugebenden Tupel sind für die Bildschirmzeile (i.a. 80 Zeichen/Zeile) zu lang

Die LIST-Ausgabe erfolgt über das Druckgerät, wenn in den bisher beschriebenen LIST-Anweisungen das Schlüsselwort LIST durch

```
ON  PRINTER
```

ergänzt wird; sonst ist das Ausgabegerät das Dialoggerät.

Mit der Ausgabe der Blocküberschrift auf dem Drucker ist ein Seitenvorschub verbunden.

Beispiel:

```
LIST  ON  PRINTER  KEMAT;
```

3.2. List-Anweisungen mit Arithmetik

Häufig sind nicht die Daten der Datenbasis als solche von Interesse sondern hieraus abgeleitete Größen (z.B. Gewichte werden in kg benötigt, während sie in der Datenbasis in g vorliegen); ebenso ist es wünschenswert, zur Tupelauswahl Arithmetik benützen zu können.

Es wird dabei unterschieden zwischen tupelorientierten arithmetischen Verknüpfungen - es werden Attributwerte jeweils eines Tupels miteinander verknüpft - und globalen Verknüpfungen - es werden Attributwerte einer Menge von Tupel zu einem neuen Wert verknüpft.

3.2.1. Arithmetik bezüglich der Ergebnisrelation

3.2.1.1. Tupelorientierte Arithmetik

In der Ergebnisrelation können Attribute ausgegeben werden, deren Werte durch arithmetische Verknüpfungen aus den Attributwerten der Tupel von rel-name (aus 3.1.2.) gewonnen werden. Hierzu wird die Definition von attr-list in 3.1.2. um folgende Produktion erweitert:

attr ::= name = arith-expr

Semantik:

Dem arithmetischen Ausdruck arith-expr wird der Name name zugeordnet. Dieser dient, ähnlich wie bei Attributumbenennungen (vergl. 3.1.2.), als Attributname (Spaltenüberschrift) der Ergebnisrelation für die Werte, die die Auswertung des Ausdrucks für jedes Tupel von rel-name ergibt.

Beispiel:

Anfrage: Für jedes Plättchen soll die Gesamtmasse Uran (UGES) in kg ausgegeben werden.

```
LIST  MATNAM,  UGESKG = UGES * 0.001  OF  KEMAT;
```

Ergebnis:

MATNAM	UGESKG
PU NP6	0.08785
PU VAK NP3	0.0
U-35% KP3	0.12470
U-NAT NB25	1.23200
UO2 NP6	0.12700
PU NPX	0.10349

3.2.1.2. Globale Arithmetik

Die Ergebnisrelation kann aus einem Tupel bestehen, dessen Attributwerte durch globale Verknüpfungen von Attributwerten einer Relation gewonnen werden. Dazu können in der Ergebnisbeschreibung Funktionsaufrufe angegeben werden. Der Benutzer wird hiermit in die Lage versetzt nach Maximum, Minimum, Summe etc. von Attributwerten einer Menge von Tupeln zu fragen.

Syntax:

```
LIST func-list  OF  rel-name  { WHERE t-qual }
func-list  ::=  name = func  {,  name = func }(n,o)
func       ::=  func-name (attr-name)
func-name  ::=  MAX | MIN | SUM | AVG | COUNT
```

Semantik:

Mit func-list werden die gewünschten globalen Verknüpfungen als Funktionen mit einem Attributnamen attr-name als Parameter spezifiziert. Mit name = func wird jedem Funktionsaufruf ein Name zugeordnet. Dieser dient, ähnlich wie bei Attributumbenennungen, als Attributname der Ergebnisrelation für den entsprechenden Funktionswert.

Ein Funktionswert func-name (attr-name) wird berechnet

- über der Menge der Attributwerte von attr-name aller Tupel der Relation rel-name, falls kein t-qual angegeben ist.
- über den Attributwerten von attr-name der Tupel, die der Qualifikation t-qual genügen.

Zur Erläuterung von Funktionen siehe auch 2.2.5.

Bespiel:

Anfrage: Wieviel Plättchen sind an allen Orten zusammen gelagert?

```
LIST  TOTAL = SUM (ANZAHL)  OF  LAGER;
```

Ergebnis:

TOTAL
77

3.2.2. Arithmetik zur Auswahl von Tupeln

Um auch bei der Tupelauswahl von Arithmetik Gebrauch machen zu können, wird die Qualifikation t-qual aus 3.1.3. um weitere Prädikate ergänzt.

3.2.2.1. Tupelorientierte Arithmetik

Syntax:

```
predicate  ::= attr-name rop arith-expr
```

Semantik:

Mit Prädikaten dieser Form läßt sich der Wert des Attributs attr-name mit dem Wert des arithmetischen Ausdrucks arith-expr vergleichen. In arith-expr werden Attributwerte jeweils eines Tupels miteinander verknüpft (vergl. Abschnitt 2.2.4).

Der Typ des Attributs attr-name muß mit dem Ergebnistyp des arithmetischen Ausdrucks arith-expr verträglich (Abschnitt 2.2.6) sein.

Bemerkung:

Dieses Prädikat ist eine Verallgemeinerung der Prädikate

```
predicate  ::= attr-name rop const
predicate  ::= attr-name-1 rop attr-name-2
```

von Abschnitt 3.1.3, da arith-expr auch eine Konstante oder ein Attributname sein kann.

Beispiel:

Anfrage: Bei welchen Plättchen macht das Isotop U235 mehr als 10 % der Gesamtmasse Uran UGES aus?

```
LIST MATNAM OF KEMAT WHERE U235 GT UGES*0.1;
```

Ergebnis:

MATNAM
U-35% KP3
PU NPX

3.2.2.2. Globale Arithmetik

Syntax:

```
predicate  ::= attr-name-1 rop func
func       ::= func-name (attr-name-2)
```

Semantik:

Es wird der Attributwert von attr-name-1 verglichen mit dem Funktionswert func-name (attr-name-2). Dieser wird berechnet über der Menge der Attributwerte zu attr-name-2 aller Tupel der Relation rel-name aus 3.1.2. Der Typ von attr-name-1 muß mit dem Typ des Funktionswertes von func verträglich (Abschnitt 2.2.6) sein.

Beispiel:

Anfrage: Welche Plättchensorte enthält am meisten Uran?

```
LIST MATNAM, UGES OF KEMAT WHERE UGES EQ MAX(UGES);
```

Ergebnis:

MATNAM	UGES
U-NAT NB25	1232.00

3.3. Enthalten-sein Prädikate

Tupelqualifikationen, wie sie bisher dargestellt wurden, erlauben nur den Vergleich von Werten aus jeweils einer Relation. Für Anfragen, bei denen zur Tupelauswahl Attributwerte einer zweiten Relation benötigt werden, stellt FAQUEL "Enthalten-sein" Prädikate (is-element-predicate) zur Verfügung. Die Tupelqualifikation t-qual aus 3.1.3 wird um solche Prädikate erweitert.

Syntax:

```
t-qual-factor        ::= is-element-predicate
is-element-predicate ::= attr-name rop
                         elem OF rel-name-2 { WHERE t-qual-2 }
elem                 ::= arith-expr | func
```

Semantik:

Durch elem OF rel-name-2 { WHERE t-qual-2 } wird in einem is-element-predicate eine zweite, "innere" Anfrage formuliert; diese liefert als Zwischenergebnis im allgemeinen eine Menge A von Attributwerten, die Ergebnisrelation der inneren Anfrage. Diese Ergebnisrelation entspricht im wesentlichen der Ergebnisrelation der LIST-Anweisung (vergl. hierzu 3.), mit der Einschränkung, daß sie nur aus einem Attribut besteht und nicht ausgegeben wird.

Für jedes Tupel t der Relation rel-name wird der Attributwert a_t von attr-name verglichen mit den Elementen $a \in A$. Ein Prädikat is-element-predicate liefert für t den Wert "wahr", wenn für mindestens einen Attributwert $a \in A$ a_t rop a gilt, sonst hat es den Wert "falsch".

Der Typ des Attributs attr-name und der Typ der Werte der Menge A müssen miteinander verträglich (2.2.6.) sein. Die Qualifikation t-qual-2 eines is-element-predicate darf keine Enthalten-sein Prädikate enthalten.

Für diese erweiterte Form von t-qual gilt nun (als Erweiterung von 3.1.3.): Ein Tupel aus rel-name erfüllt genau dann t-qual, wenn nach dem Ersetzen

- der Attributnamen in den Prädikaten predicate (Abschnitt 3.1.3) durch die entsprechenden Attributwerte des Tupels
- der Prädikate is-element-predicate durch die ihrem Wert entsprechende Boolesche Konstante TRUE, FALSE

der aus t-qual resultierende logische Ausdruck den Wert "wahr" ergibt.

Bemerkung:

In der Qualifikation t-qual können mittels is-element-predicate Attributwerte verschiedener Relationen miteinander verglichen werden (falls rel-name # rel-name-2)

Beispiel:

Anfrage: Gesucht sind die Bezeichnungen der Plättchen mit mehr als 8g U235, die sich in der Position LA/1/1/1 befinden!

```
          LIST  MATNAM  OF  KEMAT  WHERE U235 GT 8.0  AND  MATNAM EQ
(A)                                  MATNAM OF LAGER WHERE ORT EQ 'LA/1/1/1';
```

Ergebnis:

MATNAM
U-35% KP3
PU NPX

Erläuterung: Die Qualifikation dieser LIST-Anweisung enthält eine zweite, "innere" Anfrage (mit A markiert). Sie liefert als Zwischenergebnis die Menge der Namen solcher Plättchen, die in LA/1/1/1 lagern, hier also: {PU NP6, U-35% KP3, PU NPX}; der Vergleichsoperator EQ, der sich damit auf eine Menge von Werten bezieht, bedeutet in dieser Situation: "Es ist ein Element in der Menge enthalten, das gleich (EQ) dem Attribut MATNAM ist". Es werden also die Materialbezeichnungen aller KEMAT-Tupel gelistet, bei denen U235 größer als 8 und MATNAM gleich einem der drei genannten Plättchennamen ist.

3.4. Verknüpfung zweier Relationen

Mit der folgenden Art von LIST-Anweisungen ist die Formulierung von Anfragen möglich, deren Ergebnis sich aus Daten zweier verschiedener Relationen ergibt.

Syntax:

```
LIST attr-list-1  OF  rel-name-1
                JOINED TO attr-list-2  OF  rel-name-2 { BY j-qual } ;

j-qual          ::= j-qual-term | j-qual-term OR j-qual
j-qual-term     ::= j-qual-factor | j-qual-factor AND j-qual-term
j-qual-factor   ::= j-predicate | (j-qual)
j-predicate     ::= attr-name { OF rel-name-1 } rop
                                attr-name { OF rel-name-2 }
```

Semantik:

Es wird eine Ergebnisrelation definiert mit den in attr-list-1 und attr-list-2 genannten Attributen.

Die Tupel der Ergebnisrelation werden folgendermaßen gewonnen:

Fall 1: Es fehlt BY j-qual.

Jedes Tupel von rel-name-1 wird mit jedem Tupel von rel-name-2 verkettet (vergl. das Beispiel und WE/74/). Die so verketteten Tupel bilden eine Relation, die dem Kreuzprodukt (cartesisches Produkt) der beiden Relationen rel-name-1 und rel-name-2 entspricht. Die Tupel der Ergebnisrelation erhält man hieraus durch die Spaltenauswahl, die durch attr-list-1 und attr-list-2 bestimmt wird.

Fall 2: Es wird eine Joinqualifikation j-qual angegeben.
Eine Joinqualifikation j-qual besteht aus einem oder mehreren Joinprädikaten j-predicate, die durch die Booleschen Operatoren AND, OR miteinander verknüpft sind. In einem Joinprädikat j-predicate wird der Wert eines Attributs der Relation rel-name-1 mit dem Wert eines Attributs von rel-name-2 verglichen.

Dabei muß der Typ der beiden Attribute gleich sein. Die Vergleichsoperatoren rop haben die gleiche Bedeutung wie in der Tupelqualifikation t-qual (siehe 3.1.3.).

Je ein Tupel aus rel-name-1 und rel-name-2 als Tupelpaar betrachtet erfüllt genau dann j-qual, wenn nach dem Ersetzen der Attributnamen in den Joinprädikaten durch die entsprechenden Attributwerte der beiden Tupel der resultierende logische Ausdruck den Wert "wahr" ergibt.

Es werden hier nur die Tupel solcher Tupelpaare miteinander verkettet, die j-qual erfüllen. Die Tupel der Ergebnisrelation werden dann aus diesen, wie in Fall 1, durch Spaltenauswahl gewonnen.

In attr-list-1 und attr-list-2 können auch Attributumbenennungen erfolgen (vergl. hierzu 3.1.2.).

Bemerkungen:

Die Operation der Verknüpfung von Tupeln zweier Relationen unter Angabe einer Qualifikation wird in /Co 70/ als Join definiert. Durch die beliebige Verwendung der Booleschen Operatoren erlaubt FAQUEL eine Verallgemeinerung des Join.

Beispiel:

Anfrage: Zu jedem Ort sind die Verträge der Materialien anzugeben, die dort lagern!

```
LIST ORT OF LAGER   JOINED TO   VERTRG OF KEMAT
                BY MATNAM OF LAGER  EQ  MATNAM OF KEMAT;
```

Ergebnis:

ORT	VERTRG
LA/1/1/1	V1
LA/2/1/9	V1
LA/1/1/1	V3
LA/1/1/1	V6

Erläuterung:

Es wird jedes Tupel von Lager mit den Tupeln von KEMAT verkettet, die in MATNAM (der Relation KEMAT!) den gleichen Wert haben: Bild 3 zeigt, welche Tupel von LAGER mit welchen Tupeln von KEMAT verkettet sind. Aus den verketteten Tupeln werden nach Spaltenauswahl als Ergebnis die Spalten ORT und VERTRG aufgelistet.

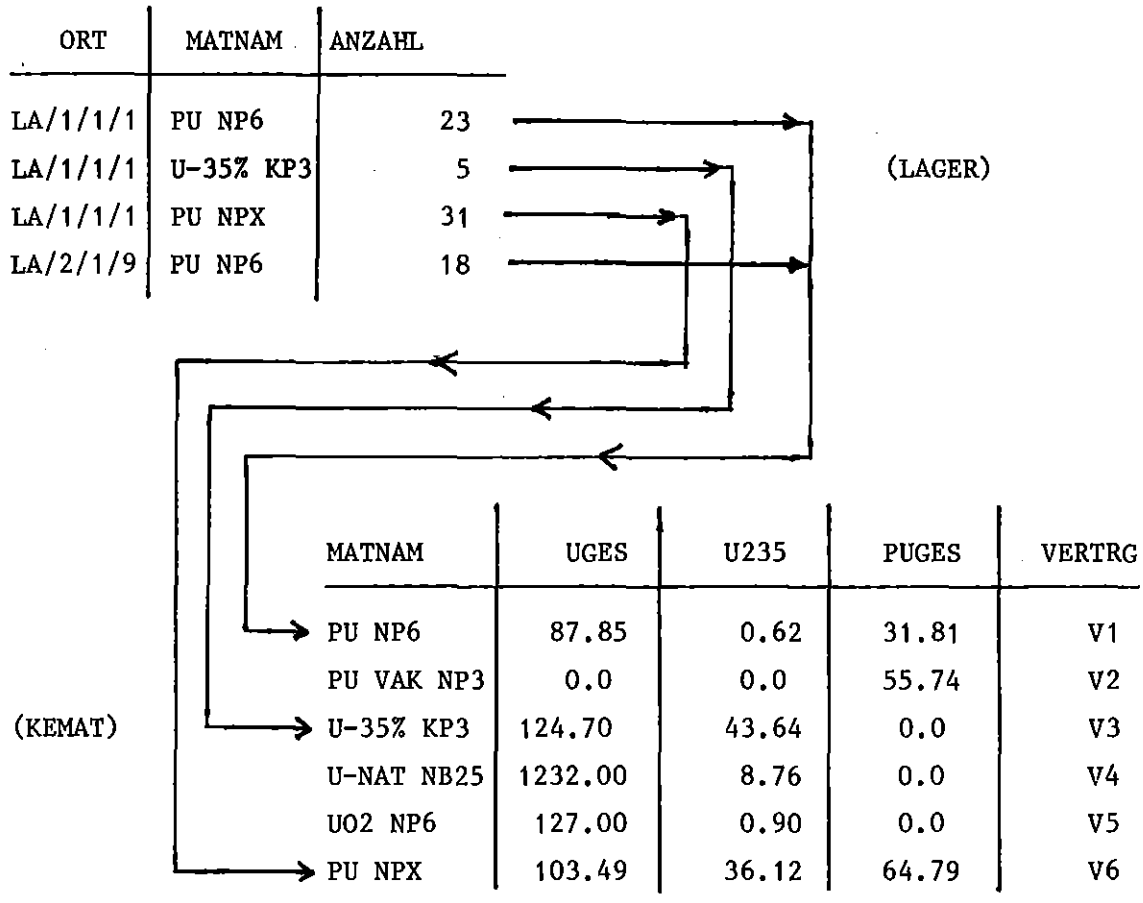


Bild 3: Beispiel zur Verknüpfung zweier Relationen

Man beachte, daß nicht jedes Tupel von KEMAT mit einem Tupel aus LAGER verkettet ist. Wären in LAGER Plättchen aufgeführt, die in KEMAT nicht enthalten sind, so gäbe es auch nicht verkettete Tupel in LAGER. (Dieser Fall ist für die Beispieldatenbank semantisch nicht sinnvoll und würde auf eine fehlerhafte Datenbank hinweisen!)

3.5. Sortieren der Ergebnisrelation

Man kann im allgemeinen nicht davon ausgehen, daß die Tupel der Ergebnisrelation in einer bestimmten Reihenfolge ausgegeben werden. Wird eine sortierte Ausgabe gewünscht, so ist dies gesondert zu spezifizieren.

Syntax:

```
LIST result ORDER BY attr-name order;
order      ::= ASC | DESC
```

Semantik:

result enthält die Ergebnisbeschreibung der LIST-Anweisung (vergl. hierzu 3.7.). Mit ORDER BY attr-name order wird angegeben, daß die Ergebnisrelation nach dem Attribut attr-name sortiert auszugeben ist. Mit ASC bzw. DESC wird auf- bzw. absteigende Sortierung spezifiziert. Das Sortieren wird auf der Basis der üblichen kleiner/gleich Beziehung durchgeführt (vergl. hierzu 2.2.7.).

Beispiel:

Anfrage: Auflisten aller Plättchensorten, die Plutonium enthalten, nach steigendem Urangehalt.

```
LIST  KEMAT  WHERE  PUGES  GT  0  ORDER  BY  UGES  ASC;
```

Ergebnis:

MATNAM	UGES	U235	PUGES	VERTRG
PU VAK NP3	0.0	0.0	55.74	V2
PU NP6	87.85	0.62	31.81	V1
PU NPX	103.49	36.12	64.79	V6

3.6. Duplikate in der Ergebnisrelation

Die Ergebnisrelation einer LIST-Anweisung enthält bei Spaltenauswahl im allgemeinen Duplikate von Tupeln; die Elimination von Duplikaten ist eine (zeit-)aufwendige Operation und wird deshalb nicht standardmäßig ausgeführt. Durch Einfügen des Schlüsselwortes UNIQUE in eine LIST-Anweisung kann das Entfernen von Duplikaten gefordert werden. Die LIST-Anweisung lautet dann:

```
LIST  UNIQUE  result;
```

Zur Definition von result siehe Abschnitt 3.7.

Beispiel:

Anfrage: Welche Lagerorte gibt es?

```
LIST  UNIQUE  ORT  OF  LAGER;
```

Ergebnis:

ORT
LA 1/1/1
LA 2/1/9

Anfrage: wie oben, jedoch mit Ergebnisausgabe auf Drucker.

```
LIST  ON  PRINTER  UNIQUE  ORT  OF  LAGER;
```

3.7. Die Syntax der LIST-Anweisung

Es folgt eine Zusammenfassung der Syntax der LIST-Anweisung, aus der insbesondere die Definition von result (Ergebnisbeschreibung, siehe 3.) zu entnehmen ist. Auf result wurde bereits in den Abschnitten 3.5 und 3.6 Bezug genommen; diese Definition von result wird in den folgenden Abschnitten nochmals benötigt.

```
LIST { ON PRINTER } { UNIQUE }  result  { ORDER BY attr-name order };

     result          ::=  {attr-list  OF}  rel-name  {WHERE  t-qual} |
                          list-name {WHERE  t-qual} |
                          func-list OF rel-name {WHERE  t-qual} |
                          attr-list OF rel-name  JOINED  TO  attr-list OF
                                                          rel-name {BY j-qual}
     t-qual          ::=  t-qual-term | t-qual-term OR t-qual
     t-qual-term     ::=  t-qual-factor | t-qual-factor AND t-qual-term
     t-qual-factor   ::=  predicate | is-element-predicate | (t-qual)
     predicate       ::=  attr-name rop arith-expr | attr-name rop func
     is-element
         -predicate  ::=  attr-name rop elem OF rel-name { WHERE t-qual } ;
     elem            ::=  arith-expr | func
     rop             ::=  EQ | NE | LT | LE | GT | GE
     attr-list       ::=  attr | attr, attr-list
     attr            ::=  {name = } attr-name | name = arith-expr
     func-list       ::=  name = func {, name = func }  n
                                                        o
     func            ::=  func-name (attr-name)
     func-name       ::=  MAX | MIN | SUM | AVG | COUNT
     j-qual          ::=  j-qual-term | j-qual-term OR j-qual
     j-qual-term     ::=  j-qual-factor | j-qual-factor AND j-qual-term
     j-qual-factor   ::=  j-predicate | (j-qual)
     j-predicate     ::=  attr-name { OF rel-name } rop attr-name { OF rel-name }
     order           ::=  ASC | DESC
```

(Zur Erläuterung von list-name siehe Abschnitt 8.1., von arith-expr siehe Abschnitt 2.2.4).

4. Update-Anweisungen

4.1. Einfügen von Tupeln

4.1.1. Einfügen von Tupeln über das Dialoggerät

Syntax:

INSERT INTO rel-name: < const { , const } $\begin{matrix} n \\ o \end{matrix}$ > ;

Semantik:

Die in < und > eingeschlossene Konstantenliste wird als Tupel aufgefaßt und in die Relation rel-name eingefügt.

Die Anzahl der Konstanten muß gleich sein der Anzahl der Attribute der Relation und der Typ der i-ten Konstanten muß mit dem Typ des i-ten Attributs der Relationenbeschreibung von rel-name (Abschnitt 8.3) verträglich sein. Es sind hierbei nur die einfachen Attribute relevant (vergl. hierzu Abschnitt 2.2.6). Die Reihenfolge der Konstanten ist also nicht beliebig!

Beispiel:

An die Position LA/2/5/2 sind 15 Plättchen PU VAK NP3 transportiert worden; es ist also ein entsprechendes Tupel in LAGER einzufügen:

INSERT INTO LAGER: < 'LA/2/5/2', 'PU VAK NP3', 15 > ;

Ergebnis: Nach dem Einfügen der in < und > eingeschlossenen Attributwerte enthält LAGER folgende Tupel:

ORT	MATNAM	ANZAHL
LA/1/1/1	PU NP6	23
LA/1/1/1	U-35% KP3	5
LA/1/1/1	PU NPX	31
LA/2/1/9	PU NP6	18
LA/2/5/2	PU VAK NP3	15

4.1.2. Einfügen einer vollständigen Relation in eine andere Relation

Syntax:

INSERT INTO rel-name-1 : rel-name-2;

Semantik:

Es werden alle Tupel der Relation rel-name-2 in die Relation rel-name-1 eingefügt. Es muß rel-name-1 # rel-name-2 sein!

Die beiden Relationen müssen vereinigungsverträglich sein, d.h. sie müssen aus gleich vielen Attributen bestehen und der Typ des i-ten Attributs von rel-name-1 muß gleich sein dem Typ des i-ten Attributs von rel-name-2. Es sind hierbei nur die einfachen Attribute relevant.

4.1.3. Spaltenauswahl beim Einfügen

Syntax:

INSERT INTO rel-name-1: attr-name {,attr-name } $^{n}_{o}$ OF rel-name-2;

Semantik:

Aus den Tupeln der Relation rel-name-2 werden neue Tupel mit den Werten der genannten Attribute attr-name zusammengestellt (Spaltenauswahl, siehe 3.1.2.) und diese in die Relation rel-name-1 eingefügt. Es muß rel-name-1 # rel-name-2 sein!

Die Menge der einzufügenden Tupel und die Relation rel-name-1, in die eingefügt wird, müssen vereinigungsverträglich (siehe 4.1.2.) sein.

Beispiel:

Die Relation LAGNAM (s. Abschnitt 2.1) enthalte noch keine Tupel. Mit

```
INSERT INTO LAGNAM: ORT OF LAGER;
```

werden alle Lagerpositionen in LAGNAM eingetragen. LAGNAM enthält dann also:

ORT
LA/1/1/1
LA/1/1/1
LA/1/1/1
LA/2/1/9

4.1.4. Tupelauswahl beim Einfügen

Syntax:

```
INSERT INTO rel-name-1 : rel-name-2 WHERE t-qual;
```

Semantik:

Es werden die Tupel der Relation rel-name-2 in die Relation rel-name-1 eingefügt, die der Qualifikation t-qual genügen. Es muß rel-name1 # rel-name-2 sein!
t-qual ist die Tupelqualifikation der LIST-Anweisung (siehe 3.7.).

rel-name1 und rel-name-2 müssen vereinigungsverträglich sein, nur die einfachen Attribute sind relevant (vergl. 4.1.2.)

4.1.5. Spalten- und Tupelauswahl beim Einfügen

Syntax:

INSERT INTO rel-name-1 : attr-name { ,attr-name } $\begin{matrix} n \\ o \end{matrix}$ OF
rel-name-2 WHERE t-qual;

Semantik:

Aus den Tupeln der Relation rel-name-2, die der Tupelqualifikation t-qual (siehe 3.7.) genügen, werden wie in 4.1.3. beschrieben neue Tupel zusammengestellt und diese in die Relation rel-name-1 eingefügt. Die Menge der einzufügenden Tupel und rel-name-1 müssen vereinigungsverträglich (4.1.2.) sein.

Beispiel:

Die Relation LAGNAM der Beispieldatenbank enthalte noch keine Tupel. Mit

INSERT INTO LAGNAM: ORT OF LAGER WHERE MATNAM EQ 'PU NP6';

werden die Positionen in LAGNAM eingetragen, an denen Plättchen vom Typ PU NP6 lagern. LAGNAM enthält dann also:

ORT
LA/1/1/1
LA/2/1/9

4.2. Löschen von Tupeln

Syntax:

```
DELETE IN rel-name { WHERE t-qual };
```

Semantik:

Bei Angabe einer Tupelqualifikation werden die Tupel der Relation rel-name gelöscht, die durch t-qual ausgewählt werden.

Fehlt t-qual, so werden alle Tupel in rel-name gelöscht; die Relation bleibt dabei bestehen, sie enthält jedoch keine Tupel mehr.

t-qual ist die Tupelqualifikation der LIST-Anweisung (vergl. 3.7.).

Beispiel:

Es seien alle Plättchen, die kein Plutonium enthalten, ausgelagert worden, so daß deren Beschreibungen nicht mehr benötigt werden. Mit folgender Anweisung kann die Relation KEMAT auf den aktuellen Stand gebracht werden:

```
DELETE IN KEMAT  WHERE PUGES EQ 0.0;
```

Ergebnis: KEMAT enthält dann noch folgende Tupel:

MATNAM	UGES	U235	PUGES	VERTRG
PU NP6	87.85	0.62	31.81	V1
PU VAK NP3	0.0	0.0	55.74	V2
PU NPX	103.49	36.12	64.79	V6

4.3. Modifizieren von Tupeln einer Relation

Syntax:

REPLACE IN rel-name: attr-name := const { ,attr-name := const } $\begin{matrix} n \\ o \end{matrix}$
{ WHERE t-qual } ;

Semantik:

Ist keine Tupelqualifikation t-qual angegeben, so werden in allen Tupeln der Relation rel-name die Attributwerte der Attribute attr-name durch die jeweiligen Konstanten const ersetzt.

Ist eine Qualifikation t-qual angegeben, so werden nur in den Tupeln, die der Qualifikation genügen, Attributwerte modifiziert.

attr-name muß ein Attribut der Relation rel-name sein.

Für jedes attr-name := const muß der Typ von attr-name mit dem Typ der Konstanten const verträglich sein.

Beispiel::

Am Ort LA/1/1/1 sind 4 Plättchen der Sorte PU NP6 dazugekommen, es befinden sich somit nicht mehr 23 sondern 27 Plättchen PU NP6 in LA/1/1/1/.

```
REPLACE IN LAGER:  ANZAHL := 27
WHERE LAGER EQ 'LA/1/1/1' AND MATNAM EQ 'PU NP6';
```

5. Der Tupelmodus

5.1. Motivation

In Abschnitt 3 wird das Datenretrieval beschrieben, wie es im sogenannten "Standardmodus", dem S-Modus, abläuft: charakteristisch für den S-Modus ist, daß der Benutzer erst nach vollständiger Ausgabe der Ergebnisrelation (d.h. nach Ausgabe des letzten Blocks von Tupeln) bzw. nach einem CANCEL-Kommando eine neue Anfrage eingeben kann.

In der Praxis gibt es jedoch Situationen, in denen es wünschenswert ist, daß der Benutzer nach Ausgabe einer jeden Zeile die Kontrolle zurückerhält und er dann - eventuell unter Berücksichtigung bisher angefallener Daten - eine neue FAQUEL-Anweisung eingeben kann.

Beispiel:

Relationen in konkreten Anwendungen können eine große Anzahl von Attributen (bis zu 40) umfassen, oder Attribute, die bei der Ausgabe lange Zeichenketten darstellen (Adressen, Buchtitel).

Es ist somit leicht möglich, daß die Zeilenlänge des Ausgabegerätes (typische Werte sind 80 bzw. 132 Zeichen pro Zeile) für die Ausgabe eines Tupels nicht ausreicht und eine übersichtliche, tabellarische Ausgabe nicht möglich ist.

Eine Lösung dieses Problems kann darin bestehen, daß der Benutzer zunächst wie in Abschnitt 3 dargestellt eine LIST-Anweisung mit einer Attributliste so formuliert, daß die Tupel der Ergebnisrelation noch in je einer Bildschirmzeile ausgegeben werden können, er aber nach Ausgabe einer Zeile in die Lage versetzt wird, nachträglich weitere Attributwerte zu diesem Tupel aus der Ausgangsrelation rel-name anzufordern.

Ein solches "tupelweises" Vorgehen kann besonders dem gelegentlichen Benutzer die Arbeit erleichtern und die Akzeptanz des Systems verbessern. Das folgende Beispiel soll dies verdeutlichen:

Die Aufgabe, alle Tupel einer Relation zu löschen, für die ein Attribut A einen Wert a und ein Attribut B einen von n Werten { b_1, ..., b_n } hat, könnte bei tupelweisem Arbeiten so gelöst werden, daß man zuerst alle Tupel mit A EQ a anfordert. Die so qualifizierten Tupel (sie sind i.a. eine Obermenge der zu löschenden Tupel) läßt der Benutzer sich nun einzeln vorlegen, und überprüft bei jedem, ob es in B einen der Werte b_i hat; falls ja wird es mit einem DELETE-Kommando gelöscht, sonst wird das nächste Tupel angefordert. Dies erspart die möglicherweise umständliche Formulierung einer Qualifikation.

A EQ a AND (B EQ b_1 OR B EQ b_2 OR .....B EQ b_n)

5.2. FAQUEL im Tupelmodus

Ein tupelweises Arbeiten im soeben geschilderten Sinn erlaubt FAQUEL im "Tupelmodus". FAQUEL läßt sich aus dem S-Modus in den Tupelmodus versetzen durch das Kommando

```
TUPLEMODE ON;
```

FAQUEL meldet sich dann mit "TM-READY", im Gegensatz zum "READY" des S-Modus.

In den Standardmodus kehrt man zurück mit dem Kommando

```
TUPLEMODE OFF;
```

FAQUEL meldet sich dann mit "READY".

5.2.1. Begriffe

Für die nachfolgende Beschreibung des Tupelmodus wird angenommen, daß sich FAQUEL im Tupelmodus befindet und gerade die Anfrage

```
LIST MATNAM, U235 OF KEMAT WHERE PUGES LE 0.0;
```

bearbeitet; diese Anfrage ist damit die aktuelle Anfrage

Das aktuelle Tupel zu einer Anfrage Q_i im Tupelmodus (im S-Modus hat dieser Begriff keinen Sinn!) ist das von FAQUEL zu Q_i zuletzt ausgegebene Tupel von rel-name.

Für die folgenden Erläuterungen wird als aktuelles Tupel zur Anfrage von oben das Tupel

```
U-NAT NB25   1232.00      8.76      0.0      V4
```

angenommen (vergl. Bild 1, Zeile 4).

Die Ausgabe am Bildschirm zu diesem Zeitpunkt ist (5.2.2., Fall 1!)

MATNAM	U235
U-NAT NB25	8.76

Im Tupelmodus kann FAQUEL mehrere Anfragen gleichzeitig bearbeiten (s. Abschnitt 5.2.4.), die hierbei referierten Relationen werden die "belegten Relationen" genannt.

5.2.2. Anfordern des nächsten Tupels

Im Tupelmodus erhält der Benutzer nach der Ausgabe eines jeden Tupels der aktuellen Anfrage die Kontrolle zurück. Er kann das nächste Tupel anfordern durch Eingabe eines Semikolon.

Fall 1: Es gibt weitere Tupel zur aktuellen Anfrage.
Es wird ein Tupel ausgegeben, versehen mit einer Überschrift, wie in Abschnitt 3 für einen Block von Tupeln beschrieben. Man vergleiche auch als Beispiel die Erläuterung zu "aktuelles Tupel" von 5.2.1.

Fall 2: Die aktuelle Anfrage ist vollständig abgearbeitet, d.h. es gibt kein Tupel (mehr) zur aktuellen Anfrage. Dies wird angezeigt durch die FAQUEL-Meldung "AKTUELLE ANWEISUNG BEENDET" (vergl. 5.2.4.).

5.2.3. Anfordern von Attributwerten des aktuellen Tupels

Mit der GIVE-Anweisung kann sich der Benutzer Attributwerte des aktuellen Tupels vorlegen lassen; sie hat die Form

GIVE attr-name { , attr-name } $\begin{matrix} n \\ o \end{matrix}$;

oder

GIVE list-name;

wobei list-name ein Bezeichner für eine Liste von Attributnamen ist (s. 8.1.)

Mit einer GIVE-Anweisung werden also die explizit oder indirekt durch list-name spezifizierten Attributwerte des aktuellen Tupels von rel-name angefordert; die Attributwerte werden ohne Überschrift ausgegeben. Zu einem aktuellen Tupel können beliebig viele GIVE-Anweisungen eingegeben werden, Attributnamen können mehrfach auftreten.

Im Beispiel von 5.2. liefert also GIVE VERTRG; den Vertrag "V4".

5.2.4. Formulierung weiterer Anfragen

Nach Ausgabe eines jeden Tupels der aktuellen Anfrage Q_i kann man eine weitere Anfrage Q_{i+1} formulieren, es sind die LIST-Anweisungen der Abschnitte 2.1, 2.2, 2.3 zugelassen. Diese wird dann zur aktuellen Anfrage und wird ebenfalls im Tupelmodus abgearbeitet. Die bis zu diesem Zeitpunkt aktuelle Anfrage Q_i wird zum Vorgänger der neuen aktuellen Anfrage Q_{i+1}. Dies kann im Prinzip beliebig häufig wiederholt werden, die vorliegende Implementierung erlaubt eine "Schachtelungstiefe" von 6 Anfragen, also 5 Vorgängeranfragen Q_1, Q_2, Q_3, Q_4, Q_5, neben der aktuellen Anfrage Q_6.

Zur Bearbeitung der Vorgängeranfrage einer aktuellen Anfrage kehrt man zurück

a) durch Eingabe des Kommandos

CANCEL;

Hiermit wird (vergl. CANCEL; von Abschnitt 3) die Bearbeitung der aktuellen Anfrage Q_{i+1} abgebrochen.

b) implizit nach Ausgabe des letzten Tupels zur Anfrage Q_{i+1}.

In beiden Fällen wird die Rückkehr zur Vorgängeranfrage Q_i durch die FAQUEL-Meldung

"AKTUELLE ANWEISUNG BEENDET"

angezeigt (vergl. Fall 2 von 5.2.2.), Q_i ist damit wieder die aktuelle Anfrage. Gibt es keine Vorgängeranfrage mehr, ist also Q_1 ausgeführt, so wird dies durch die zusätzliche FAQUEL-Meldung "TM-READY" angezeigt. Das aktuelle Tupel von Q_i wird durch die Anfragen Q_{i+1}, Q_{i+2}, etc. nicht berührt, insbesondere also beziehen sich GIVE-Anweisungen unmittelbar vor der Eingabe von Q_{i+1} und unmittelbar nach der Beendigung von Q_{i+1} auf das gleiche aktuelle Tupel.

5.2.5. Update-Anweisungen im Tupelmodus

Neben retrieval-Operationen erlaubt FAQUEL im Tupelmodus auch das Modifizieren von Relationen und Tupeln.

5.2.5.1. Analog zu 5.2.4. kann während der Bearbeitung der aktuellen Anfrage eine der update-Anweisungen von Abschnitt 4 eingegeben werden. Diese werden wie im Standardmodus bearbeitet, die Ausführung wird durch die FAQUEL-Meldung

"AKTUELLE ANWEISUNG BEENDET"

angezeigt.

Einschränkung: update-Anweisungen dürfen sich nicht auf belegte Relationen beziehen (insbesondere also auch nicht auf die Relation der aktuellen Anfrage Q_i)

5.2.5.2. Besondere Möglichkeiten zum Modifizieren der Relation der aktuellen Anfrage bietet FAQUEL, indem es dem Benutzer möglich ist

- das aktuelle Tupel zu löschen mit

  DELETE;

  Dies bewirkt zugleich das Anfordern des nächsten Tupels (s. 5.2.2).

- ein neues Tupel in die Relation der aktuellen Anfrage einzufügen, bei dem einige Attributwerte mit Attributen des aktuellen übereinstimmen und andere explizit eingegeben werden können. Dies erreicht man mit Anweisungen der Form

  INSERT attr-name := const { ,attr-name := const } $^{n}_{o}$;

Mit

```
INSERT MATNAM := 'U-NAT  NB 26', PUGES := 1.61, VERTRG := 'V9';
```

wird das Tupel

```
< 'U-NAT  NB26', 1232.00, 8.76, 1.61, 'V9' >
```

in KEMAT eingefügt: die nicht explizit spezifizierten Attributwerte 1232.00 (Attribut UGES) und 8.76 (Attribut U235) sind identisch mit denen des aktuellen Tupels. Das aktuelle Tupel bleibt davon unberührt. Wie bei den INSERT-Anweisungen von Abschnitt 4.1 ist auf Verträglichkeit der Konstanten und Attribute zu achten

- Attributwerte des aktuellen Tupels zu ändern.
  Dies ist die Funktion von Anweisungen der Form

```
REPLACE attr-name := const { ,attr-name := const }  n
                                                    o ;
```

Eine Anweisung

```
REPLACE A1 := a1, A2 := a2, ... ;
```

ist äquivalent zur Anweisungsfolge

```
INSERT A1 := a1, A2 := a2, .. ;
DELETE;
```

d.h. das aktuelle Tupel wird in der Relation ersetzt durch ein Tupel, das sich nur in den spezifizierten Attributen vom aktuellen Tupel unterscheidet, und das nächste Tupel angefordert.

Die Konstanten müssen mit dem jeweiligen Attribut typverträglich sein.

Einschränkung: Diese update-Anweisungen sind nur zulässig, wenn die Relation der aktuellen Anfrage nicht durch eine Vorgängeranfrage belegt ist. (Diese Einschränkung ist also schwächer als die von 5.2.5.1.)

6. Zwischenrelationen

FAQUEL wurde so definiert, daß kompliziert geschachtelte Anfragen vermieden werden (vergl. hierzu 1.3). Ab einem gewissen Grad von Komplexität muß daher eine Anfrage in Einzelschritte, d.h. mehrere FAQUEL-Anweisungen zerlegt werden. Jeder Schritt liefert ein Zwischenergebnis, welches in den nachfolgenden Anweisungen verwendet wird. Diese Zwischenergebnisse werden mit FAQUEL in temporären Relationen, sog. Zwischenrelationen, abgelegt.

Zwischenrelationen können auch dazu verwendet werden, Zwischenergebnisse bereitzustellen, die dann wiederholt in mehreren aufeinanderfolgenden Anweisungen benötigt werden. Die Benutzung von Zwischenrelationen kann somit zu einem effizienteren Arbeiten beitragen.

Zwischenrelationen können in allen FAQUEL-Anweisungen wie "normale" Relationen verwendet werden; sie werden jedoch beim Beenden des FAQUEL-Systems automatisch gelöscht (vergl. hierzu 6.2.).

6.1. Definition von Zwischenrelationen

Syntax:

```
LET  inter-rel-name BE result;
LET  inter-rel-name BE < const-list > ;
const-list  ::=  name = const { , name = const } n
                                                 o
```

Semantik:

Es wird eine Zwischenrelation angelegt mit dem Namen inter-rel-name. Dieser darf nicht identisch sein mit dem Namen einer schon bestehenden Relation oder Zwischenrelation.

Fall 1:

result ist definiert wie in Abschnitt 3.7 und beschreibt wie bei den LIST-Anweisungen von 3.7. eine Ergebnisrelation, die nun als Zwischenrelation definiert wird. Die Attributnamen und -typen der Zwischenrelation sind die der Ergebnisrelation. Ist mit result eine Verknüpfung von Relationen (3.4.) spezifiziert, so ist darauf zu achten, daß die Attributnamen der Zwischenrelation eindeutig sind. Die Eindeutigkeit ist evtl. mittels Attributumbenennungen (name = attr-name, vergl. 3.1.2.) herzustellen.

Fall 2:

Bei Angabe einer Konstantenliste const-list besteht die Zwischenrelation nur aus einem Tupel. Die Attributnamen sind durch name definiert, die Attributtypen durch die Typen der Konstanten von const-list. Attributwerte des Tupels sind die Konstanten aus const-list.

Beispiel:

Im Verlauf der Dialogsitzung werden mehrfach die Daten der Plättchen benötigt, die am Ort LA/1/1/1 liegen. Um diese Plättchen nicht in jeder der Anweisungen neu qualifizieren zu müssen, werden sie in einer Zwischenrelation LAGER1 zur Verfügung gestellt:

```
LET LAGER1 BE MATNAM, ANZAHL OF LAGER WHERE ORT EQ 'LA/1/1/1';
```

6.2. Verwaltung von Zwischenrelationen

Mit der Anweisung

```
CLEAR inter-rel-name;
```

kann die Zwischenrelation inter-rel-name explizit (also bereits während der Dialogsitzung) gelöscht werden.

Mit der Anweisung

```
INTERMEDIATE;
```

kann sich der Benutzer eine Liste mit den Namen aller von ihm im Verlauf der Sitzung definierten und noch nicht gelöschten Zwischenrelationen ausgeben lassen.

7. Gruppierung

Mit den LIST-Anweisungen von Abschnitt 3 kann man sich aus einer Datenbasis Attributwerte zu bestimmten Tupeln ausgeben lassen. Es kann jedoch auch der Wunsch auftreten, Informationen über bestimmte Tupelmengen abzufragen. Eine Anfrage dieser Art an die Beispieldatenbasis könnte sein:

Wieviel Plättchen sind an jedem Ort gelagert?

Gefragt wird also nach der Summe der Plättchenanzahlen an den einzelnen Orten. Diese Anfrage könnte man so beantworten, daß man sich die Relation LAGER in Gruppen von Tupeln aufgeteilt denkt, die definiert sind durch den gleichen Wert im Attribut ORT; für jede Gruppe wird dann die Summe über die Werte des Attributs ANZAHL gebildet und ausgegeben.

Zur Realisierung von Anfragen dieser Art dient in FAQUEL das Konzept der Gruppierung. Es stellt Sprachelemente zur Verfügung, eine Relation in Gruppen aufzuteilen und aus einer solchen "gruppierten" Relation mittels einer "Gruppenqualifikation" Gruppen auszuwählen. In den folgenden Abschnitten werden die Gruppierung sowie das Retrieval aus gruppierten Relationen erläutert.

7.1. Gruppierte Relationen

Syntax:

LET inter-rel-name BE result GROUP BY attr-name;

Semantik:

result ist definiert wie in Abschnitt 3.7 und beschreibt eine Ergebnisrelation, die durch GROUP BY attr-name in Gruppen von Tupeln aufgeteilt und als Zwischenrelation inter-rel-name (vergl. 6.1.) abgelegt wird. attr-name ist das "Gruppierungsattribut". Jede Gruppe besteht jeweils aus den Tupeln von result, die im Gruppierungsattribut übereinstimmen. result wird damit in disjunkte Teilmengen von Tupeln aufgeteilt.

Beispiel:

Mit

```
LET GLAGER BE LAGER GROUP BY ORT;
```

wird die Relation LAGER nach dem Attribut ORT gruppiert. Die Relation GLAGER besteht aus 2 Gruppen: die erste Gruppe enthält alle Tupel mit dem Attributwert LA/1/1/1 die andere enthält alle Tupel mit dem Attributwert LA/2/1/9.

7.2. Retrieval bei gruppierten Relationen

7.2.1. Ausgabe aller Gruppen

Syntax:

```
LIST GROUPED  group-result;

group-result ::= gr-attr-list OF inter-rel-name
gr-attr-list ::= gr-attr | gr-attr, gr-attr-list
gr-attr      ::= { name = } attr-name | name = func
```

Semantik:

Die Relation inter-rel-name muß eine gruppierte Relation sein. group-result spezifiziert die Ergebnisrelation der LIST GROUPED-Anweisung. Sie enthält zu jeder Gruppe der Relation inter-rel-name ein Tupel mit den durch gr-attr-list spezifizierten Informationen, nämlich Attributwerte (attr-name) bzw. Funktionswerte (func). Die Attributnamen der Ergebnisrelation sind gegeben durch attr-name bzw. name (vergl. hierzu 3.1.2., Attributumbenennungen). Die Attributwerte eines Tupels können sein:

- der Wert des Gruppierungsattributes attr-name der betreffenden Gruppe
- Funktionswerte func, berechnet über den Attributwerten der betreffenden Gruppe

Beispiel:

Zur Beantwortung der Anfrage

Wieviele Plättchen sind an jedem Ort gelagert?

sei (wie im Beispiel von Abschnitt 7.1 gezeigt) die Relation LAGER schon nach dem Attribut ORT gruppiert und als Zwischenrelation GLAGER abgelegt.

Mit

```
LIST GROUPED ORT, SUMME = SUM (ANZAHL) OF GLAGER;
```

wird für jede Gruppe von GLAGER, d.h. für jeden Ort, die Summe der Plättchenanzahlen ermittelt und zusammen mit der Ortsbezeichnung ausgegeben.

ORT	SUMME
LA/1/1/1	59
LA/2/1/9	18

7.2.2. Auswahl von Gruppen

Syntax:

```
LIST GROUPED group-result;

group-result   ::= gr-attr-list OF inter-rel-name WHEN g-qual
g-qual         ::= g-qual-term | g-qual-term OR g-qual
g-qual-term    ::= g-qual-factor | g-qual-factor AND g-qual-term
g-qual-factor  ::= g-predicate | (g-qual)
g-predicate    ::= attr-name rop const | func rop const |
                   func-1 rop func-2 | set-comp
```

Semantik:

Mit der Gruppenqualifikation g-qual können bestimmte Gruppen der gruppierten Relation inter-rel-name ausgewählt werden. Es werden nur in den Gruppen, die g-qual genügen, die in gr-attr-list (7.2.1.) spezifizierten Gruppeneigenschaften ausgegeben.

Eine Gruppenqualifikation g-qual besteht aus einem oder mehreren Gruppenprädikaten g-predicate, die durch die Booleschen Operatoren AND, OR miteinander verknüpft werden können.

Eine Gruppe aus inter-rel-name erfüllt genau dann g-qual, wenn nach dem Ersetzen der Attributnamen, Funktionen und Mengen in den Gruppenprädikaten durch die entsprechenden Attributwerte, Funktionswerte und Mengen von Attributwerten (7.2.3.) einer Gruppe der resultierende logische Ausdruck den Wert "wahr" ergibt.

a) In einem Gruppenprädikat der Art

```
attr-name  rop  const
```

muß attr-name das Gruppierungsattribut sein. Der Typ der Konstanten muß mit dem Typ des Attributs verträglich (2.2.6.) sein.

b) Die Funktion func eines Gruppenprädikates verknüpft die Attributwerte jeweils einer Gruppe zu einem Funktionswert.

In einem Gruppenprädikat der Art

func rop const

wird für jede Gruppe der Funktionswert von func mit der Konstanten const verglichen. Der Typ des Funktionswertes (vergl. 2.2.5.) muß mit dem Typ der Konstanten verträglich (2.2.6.) sein.

Bei

func-1 rop func-2

werden für jede Gruppe die beiden Funktionswerte func-1 und func-2 miteinander verglichen. Die Typen der Funktionswerte (2.2.5.) müssen miteinander verträglich (2.2.6.) sein.

Die Prädikate set-comp dienen dem Vergleich von Tupelmengen und werden im nächsten Abschnitt erläutert.

Beispiel:

Anfrage: An welchen Orten sind mindestens 50 Plättchen gelagert?

Die Relation LAGER sei (wie im Beispiel von Abschnitt 7.1. gezeigt) als Zwischenrelation GLAGER nach dem Attribut ORT gruppiert. Zur Beantwortung der Anfrage ist somit für jede Gruppe über die Plättchenanzahl zu summieren. Aufzulisten sind dann die Orte, bei denen die Summe der Plättchenanzahlen größer/gleich 50 ist. Dies wird erzielt mit:

```
LIST GROUPED ORT OF GLAGER WHEN SUM(ANZAHL) GE 50;
```

Ergebnis:

ORT
LA/1/1/1

7.2.3. Das Gruppenprädikat set-comp

```
set-comp ::= set-1  set-rop  set-2
set-1    ::= SET (attr-name { , attr-name } n o OF GROUPED inter-rel-name)
set-2    ::= SET ( { attr-name { ,attr-name } n o OF } rel-name
                   { WHERE t-qual } ) |
             SET ( < const { , const } n o > )| SET (EMPTY)
set-rop  ::= EQUAL | NOT-EQUAL | CONTAINS | IS-PART-OF
```

Semantik:

SET ist ein Operator zur Definition von Tupelmengen im mathematischen Sinne, insbesondere entfernt er eventuell auftretende Duplikate.

Mit den Prädikaten

```
set-1  set-rop  set-2
```

werden Mengenvergleiche spezifiziert mit folgender Bedeutung:

set-rop	Notation der Mengenlehre, Bedeutung
EQUAL	= , Gleichheit von set-1 und set-2
NOT-EQUAL	# , Ungleichheit von set-1 und set-2
CONTAINS	$\supseteq$, set-1 enthält set-2
IS-PART-OF	$\subseteq$, set-1 ist Teilmenge von set-2

Die Menge set-2 kann sein (sie entspricht const in den Prädikaten von Abschnitt 3.1.3.):

- set-2 ::= SET ({attr-name { , attr-name } n o OF } rel-name { WHERE t-qual })

  Die Tupel von set-2 bestehen aus den vollständigen Tupeln von rel-name oder aus einer Spaltenauswahl (3.1.2.). Außerdem können durch die Qualifikation t-qual bestimmte Tupel ausgewählt werden.

- set-2 ::= SET (< const { ,const } $\overset{n}{o}$ >)

  set-2 besteht aus einem einzigen Tupel mit den angegebenen Konstanten const als Attributwerten.

- set-2 ::= SET (EMPTY)

  set-2 ist die leere Menge

Eine Gruppe von inter-rel-name erfüllt

set-1 rop set-2,

wenn die durch

set-1 ::= SET (attr-name { ,attr-name } $\overset{n}{o}$ OF GROUPED inter-rel-name)

zu ihr definierte Menge set-1 in der durch set-rop geforderten Beziehung zu set-2 steht.

Diese (Tupel-) Menge set-1 wird für jede Gruppe der Relation inter-rel-name gebildet. Die Tupel bestehen aus den Attributwerten, die durch die Liste der Attribute attr-name spezifiziert sind.

Die beiden Tupelmengen set-1 und set-2 eines Gruppenprädikats set-comp müssen vereinigungsverträglich (s. 3.1.2.) sein. Die leere Menge SET (EMPTY) ist mit jeder Menge vereinigungsverträglich.

Die Mengenvergleiche set-comp können sowohl miteinander als auch mit anderen Gruppenprädikaten durch Boolesche Operatoren verknüpft werden (vergl. Syntax von 7.2.2.).

Beispiel:

Anfrage: Gesucht sind die Orte, bei denen alle Plättchen Plutonium enthalten.

Zunächst wird die Relation LAGER nach dem Attribut ORT gruppiert. Da für die Anfrage nur die beiden Attribute ORT und MATNAM benötigt werden, wird die Zwischenrelation GLAGER auch nur mit diesen beiden Attributen definiert.

```
LET GLAGER BE ORT, MATNAM OF LAGER GROUP BY ORT;
```

Für jede Gruppe in GLAGER, also für jeden Ort wird die Menge der Plättchennamen (set-1) gebildet. Das sind Tupel bestehend aus dem Attributwert von MATNAM.

Aus der Relation KEMAT werden die Plättchen bestimmt, die Plutonium enthalten (PUGES NE 0). Davon wird die Menge von Tupeln bestehend nur aus dem Attribut MATNAM gebildet (set-2).

Für jede Gruppe aus GLAGER wird ermittelt, ob die Menge ihrer Plättchennamen eine Teilmenge ist der Menge von Plättchennamen aus KEMAT mit PUGES NE 0. Falls ja, wird die Ortsbezeichnung ausgegeben.

Dies erhält man mit der Anweisung:

```
LIST ORT OF GLAGER WHEN
SET(MATNAM OF GROUPED GLAGER) IS-PART-OF
                              SET(MATNAM OF KEMAT WHERE PUGES NE 0);
```

Ergebnis: In der Beispieldatenbasis gibt es keinen Ort, der die Bedingung erfüllt!

8. Verschiedenes

8.1. Definition von Attributlisten

Wird im Verlauf einer Sitzung häufig eine Liste von Attributnamen einer Relation benötigt, so kann sich der Benutzer eine "Attributliste" definieren und diese in FAQUEL-Anweisungen als Ersatz für die einzelnen Attributnamen benutzen. Durch die Verwendung von Attributlisten wird der Schreibaufwand verringert und FAQUEL-Anweisungen vereinfacht.

Syntax:

```
ATTRIBUTELIST list-name = attr-name { ,attr-name } OF rel-name;
```

Semantik:

Es wird eine Attributliste mit dem Namen list-name definiert. Sie besteht aus den Attributnamen attr-name der Relation rel-name.

Beispiel:

Die Attribute UGES, U235, PUGES von KEMAT werden zu einer Liste mit dem Namen MASSEN zusammengefaßt durch:

```
ATTRIBUTELIST MASSEN = UGES, U235, PUGES OF KEMAT;
```

Die Anfragen

```
LIST UGES, U235, PUGES OF KEMAT;
LIST MASSEN;
```

sind dann gleichbedeutend.

8.2. Kardinalität einer Relation

Syntax:

CARD OF rel-name;

Semantik:

Als Ergebnis wird die Anzahl der Tupel der Relation rel-name ausgegeben.

Eine Anfrage an die Datenbasis kann darin bestehen, zu erfragen, ob ein bestimmtes Datum in ihr enthalten ist, ohne dieses auszugeben. Für Anfragen dieser Art steht folgende Anweisung zur Verfügung:

Syntax:

CHECK rel-name IF t-qual;

Semantik:

Die Relation rel-name wird daraufhin überprüft, ob sie mindestens ein Tupel enthält, welches der Qualifikation t-qual (s. 3.7.) genügt. Existiert ein solches Tupel, wird JA ausgegeben sonst NEIN.

8.3. Relationenbeschreibung

Syntax:

```
DESCRIPTION rel-name;
```

Semantik:

Zur Relation rel-name werden die Namen und Typen ihrer Attribute ausgegeben, insbesondere wird angezeigt, welche Attribute komplexe Attribute sind. Weiterhin wird die Reihenfolge der Attribute angegeben, die für die Anweisungen in Abschnitt 4 relevant ist.

8.4. Protokollierung

Syntax:

```
PROTOCOL  ON;
PROTOCOL OFF;
```

Semantik:

Mit PROTOCOL ON; wird die Protokollierung des Dialogs eingeschaltet. Jede Ein- und Ausgabe, die am Dialoggerät erfolgt, wird zusätzlich auf einem Drucker ausgegeben. Mit PROTOCOL OFF; wird die Protokollierung ausgeschaltet.

8.5. Prozeduren

FAQUEL-Anweisungen, die immer wieder benötigt werden, können als Prozeduren definiert werden. Sie erhalten einen Prozedurnamen und bleiben auch nach Beenden des FAQUEL-Systems erhalten. Zur Ausführung genügt dann ein Aufruf mit Angabe des Prozedurnamens. In Prozeduren können Konstante als formale Parameter definiert werden.

8.5.1 Prozedurdefinition

Syntax:

```
DEFINE PROCEDURE proc-name { (? par-name { , ? par-name } )} ;
    ..........
    FAQUEL-Anweisungen
    ..........
ENDPROC;
```

Semantik:

Die Prozedurdefinition wird mit der DEFINE-Anweisung eingeleitet und abgeschlossen mit ENDPROC;

In der DEFINE-Anweisung wird der Prozedurnamen proc-name angegeben und gegebenenfalls die Liste der formalen Parameter. Die FAQUEL-Anweisungen zwischen DEFINE und ENDPROC bilden den Prozedurrumpf.

8.5.2 Prozeduraufruf

Syntax:

```
EXECUTE proc-name { (const {, const } ) } ;
```

Semantik:

Mit EXECUTE ruft man die Prozedur proc-name auf. Das bedeutet, daß die Anweisungen des Prozedurrumpfes ausgeführt werden. Sind bei der Prozedurdefinition formale Parameter angegeben, so ist beim Aufruf eine entsprechende Liste von aktuellen Parametern (Konstanten const) anzugeben. Vor Ausführung der Prozedur werden die formalen Parameter im Prozedurrumpf durch die aktuellen textuell in der Reihenfolge ihrer Aufschreibung ersetzt. Dabei ist darauf zu achten, daß nach dem Ersetzen der formalen Parameter durch die Konstanten korrekte FAQUEL-Anweisungen vorliegen, z.B. ist auf Typverträglichkeit (2.2.6.) zu achten.

Beispiel:

Für Fragen der Art

An welchen Orten sind mindestens 50 Plättchen gelagert?

sind wie in Abschnitt 7 gezeigt, 2 Anweisungen erforderlich. Diese können zu einer Prozedur MATANZ zusammengefaßt werden mit einem formalen Parameter N an Stelle der Konstanten (50).

```
DEFINE PROCEDURE MATANZ (?N);
LET GLAGER BE LAGER GROUP BY ORT;
LIST GROUPED ORT OF LAGER WHEN SUM (ANZAHL) GE ?N ;
ENDPROC;
```

Die Orte, an denen mindestens 10 Plättchen gelagert sind, erhält man damit durch:

```
EXECUTE  MATANZ (10);
```

Ergebnis:

ORT
LA/1/1/1
LA/2/1/9

8.5.3. Verwaltung von Prozeduren

Löschen einer Prozedurdefinition:

```
DROP proc-name;
```

Ausgabe von Prozedurdefinitionen:

```
PROCLIST { proc-name } ;
```

Fall 1: Es ist kein Prozedurname angegeben; dann wird zu allen bisher definierten Prozeduren der Prozedurname und gegebenenfalls die Liste der formalen Parameter ausgegeben.

Fall 2: Es ist ein Prozedurname proc-name angegeben; dann werden zu proc-name der Prozedurname, gegebenenfalls die formalen Parameter und der Prozedurrumpf ausgegeben.

9. FAQUEL und die relationale Vollständigkeit

Die von CODD in /Co 71/ definierte "relationale Vollständigkeit" ist ein Maß für das Auswahlvermögen einer Anfragesprache eines relationalen Datenbanksystems. In diesem Abschnitt wird gezeigt, inwieweit FAQUEL relational vollständig ist. Um die relationale Vollständigkeit einer Sprache nachzuweisen, ist zu zeigen, daß man mit ihr aus einer Datenbasis mit Relationen in erster Normalform /We74/ jede Ergebnisrelation extrahieren kann, welche mit den Operationen der Relationenalgebra /Co 71/ erzeugt werden kann.

Es ist also nachzuweisen, daß

a) die Grundoperationen Projektion, Restriktion, Join, Division, Durchschnitt, Differenz, Vereinigung in der Sprache formulierbar sind

b) eine beliebige Hintereinanderausführung dieser Operationen möglich ist.

Für die folgende Untersuchung wird vorausgesetzt, daß die Relationen keine Duplikate enthalten (dies wird durch den FADABS-Nucleus nicht automatisch gewährleistet) und Attribute nicht weiter zerlegbar sind (erste Normalform!).

9.1. Grundoperationen

Es werden die Formulierungen der Grundoperationen in FAQUEL-Syntax angegeben. Wo Einschränkungen auftreten, werden diese erläutert.

R sei eine Relation mit m Attributen A1,A2,...,Am

Projektion:

```
LIST UNIQUE A1,...,Ak OF R;
```

Restriktion:

```
LIST UNIQUE A1,A2,...,Am OF R  WHERE t-qual;
```

(Mit der Tupelqualifikation t-qual (vergl. 3.7.) lassen sich durch die Verwendung Boolescher Operatoren allgemeinere Bedingungen formulieren als mit einer "einfachen" Restriktion nach /Co 71/.)

S sei eine Relation mit n Attributen B1,B2,...Bn

Join:

```
LIST UNIQUE A1,...,Am OF R JOINED TO B1,...,Bn OF S BY j-qual;
```

j-qual ist definiert wie in Abschnitt 3.4.

Division:

Die Division ist in FAQUEL nur eingeschränkt möglich. Zu Erläuterung der Einschränkung wird zunächst die Definition der Division nach Codd angegeben.

Zur Definition wird der Begriff der Bildmenge benötigt. Sei T (X,Y) eine binäre Relation mit den Tupel (x,y) ε T. Man versteht unter der Bildmenge von x unter T die Menge

$$gT(x) = \{ y \mid (x,y) \varepsilon T \}$$

R sei wieder eine Relation mit m Attributen, r ein Tupel aus R

r(j) ist j-te Komponente von r j = 1,2,...,m
A = (j1,...,jk) ist eine Folge aus 1,2,...,m
$\overline{A}$ ist das Komplement von A
r(A) = (r(j1),...,r(jk)) ist ein Tupel mit k Attributen

S sei eine Relation mit n Attributen, B eine Folge aus 1,2,...,n

Division ist nun definiert als

$$R(A \div B)S = \{ r(\overline{A}) \mid r \in R \text{ und } S(B) \subseteq gR(r(\overline{A})) \}$$

Die Einschränkung in FAQUEL besteht darin, daß $\overline{A}$ nur aus einer Zahl (nicht aus einer Folge!) bestehen kann, d.h. das Argument in gR ist ein einziges Attribut: In FAQUEL wird die Division über das Gruppierungskonzept und mit Hilfe der Mengenvergleiche formuliert. Die Einschränkung kommt dadurch zustande, daß eine Relation mit FAQUEL nur jeweils nach einem Attribut gruppiert werden kann und nicht nach mehreren Attributen gleichzeitig. (Gruppierung nach mehreren Attributen würde bedeuten, daß innerhalb der einzelnen Gruppen wieder neu nach einem weiteren Attribut gruppiert werden müßte, etc.)

Die eingeschränkte Division in FAQUEL lautet:

Sei A = 1,2,...,k-1,k+1,...,m $\overline{A}$ = k
LET GR BE R GROUP BY Ak;
LIST Ak OF GR
WHEN SET(A1,...,Ak-1,Ak+1,...,Am OF GR) CONTAINS SET(B1,...,Bn OF S);

Die Ergebnisrelation besteht also immer nur aus dem Gruppierungsattribut.

Aus folgenden Gründen wird diese Einschränkung der Division für vertretbar und sinnvoll gehalten:
Alle Beispiele zur Division, die in der Literatur über relationale Anfragesprachen zu finden waren und die sich auf Datenbasen mit einer Bedeutung bezogen (also nicht "konstruiert" sind) lassen sich mit der eingeschränkten Division formulieren. Das soll nicht heißen, daß keine sinnvollen Beispiele denkbar wären; dem Entwurf von FAQUEL liegt die Ansicht zugrunde, daß die Anfragen der Praxis sich mit dieser "eingeschränkten Division" erledigen lassen.

Die Division ist für die Realisierung eine sehr zeitaufwendige Operation, so daß bei der Implementierung auf einem Kleinrechner diese Einschränkung, die zudem keine wesentliche Einschränkung des Auswahlvermögens der Sprache bedeutet, gerechtfertigt erscheint.

Mengenoperationen:

FAQUEL bietet keine Operatoren für die Mengenoperationen an. Die Operation Durchschnitt läßt sich jedoch durch einen Join und anschließende Projektion formulieren und die Differenz durch einen Join und anschließende Division /Pe 75/ (die eingeschränkte Division hat eine eingeschränkte Differenz zur Folge). R und S müssen vereinigungsverträglich sein (s.4.1.2.).

Durchschnitt $R \cap S$:

```
LET HREL BE A1,...,Am OF R JOINED TO B1,...,Bm OF S
                                    BY A1 EQ B1 AND ... AND Am EQ Bm;
LIST A1,...,Am OF HREL;
```

Differenz R \ S:

Einschränkung: R und S dürfen nur aus einem Attribut bestehen

Es sei A ein Attribut von R und B ein Attribut von S

```
LET HREL BE A OF R JOINED TO B OF S BY A NE B GROUP BY A;
LIST A OF HREL WHEN SET( B OF GROUPED HREL ) CONTAINS SET(B OF S );
```

Durch den Verzicht auf eigene Operatoren für Durchschnitt und Differenz wird somit das Auswahlvermögen der Sprache nicht wesentlich eingeschränkt. Allerdings sind die Formulierungen über Join und Division schwer verständlich. Ergibt sich aus einem Anwendungsbereich heraus die Forderung nach Mengenoperationen, so sind spezielle Operatoren dafür vorzuziehen. Aus den Einsatzbereichen für FAQUEL ergab sich diese Forderung nicht.

Vereinigung R∪S:

Nach PECHERER /Pe 75/ gehört die Vereinigung von Relationen zu update-Operationen. Sie kann mittels eines INSERT-Operators erfolgen. Auf diese Weise kann auch in FAQUEL eine Vereinigung realisiert werden.

```
LET HREL BE R;
INSERT INTO HREL: S;
LIST UNIQUE HREL;
```

9.2. Hintereinanderausführung von Operationen

Die Zahl der Operationen, die sich gleichzeitig in einer einzelnen FAQUEL-Anweisung ausdrücken lassen, ist stark beschränkt. Dies ergibt sich aus dem Ziel, eine Sprache zu schaffen, die kompliziert geschachtelte, schwer verständliche Anweisungen vermeidet (s. 1.3.). Durch das Konzept der Zwischenrelationen, in denen beliebig viele Zwischenergebnisse abgelegt und in nachfolgenden Anweisungen weiterverwendet werden können, ist die Hintereinanderausführung beliebig vieler Operationen in mehreren FAQUEL-Anweisungen prinzipiell gewährleistet. Eine Beschränkung bezüglich der Anzahl der möglichen Zwischenrelationen wird sich natürlich aus der Implementierung ergeben.

10. Zur Implementierung von FAQUEL

FAQUEL wird durch ein Programmsystem /Tr 80/ realisiert, welches im wesentlichen zwei Aufgaben erfüllt:

- Verarbeitung der FAQUEL-Anweisungen
- Steuerung des Dialogablaufs für Standard- und Tupelmodus

Die Verarbeitung von FAQUEL-Anweisungen erfolgt durch eine Interpretation in zwei Schritten:

1. Analyse und Übersetzung der Anweisungen durch einen Übersetzer in eine interne Form, das sog. "Interpreterprogramm"
2. Ausführung des Interpreterprogrammes durch einen Interpreter.

Dabei wird für den Zugriff auf die Datenbank und das Datenbankschema von der FADABS-Schnittstelle EDBL /KP 79/ Gebrauch gemacht.

Die Steuerung des Dialogablaufs erfolgt durch einen Steuermodul, der die Eingabe, Übersetzung und Ausführung der Anweisungen in der Form anstößt und überwacht, wie dies zur Realisierung von Standard- und Tupelmodus erforderlich ist.

Die ausführliche Beschreibung und Dokumentation des FAQUEL-Programms findet sich in /Tr 80/.

10.1. Systemstruktur

Das Programmsystem besteht aus folgenden Hauptkomponenten:

Steuermodul
Eingabemodul
Übersetzer
Interpreter

Der Systemaufbau mit Aufrufstruktur und Datenfluß ist Bild 4 zu entnehmen.

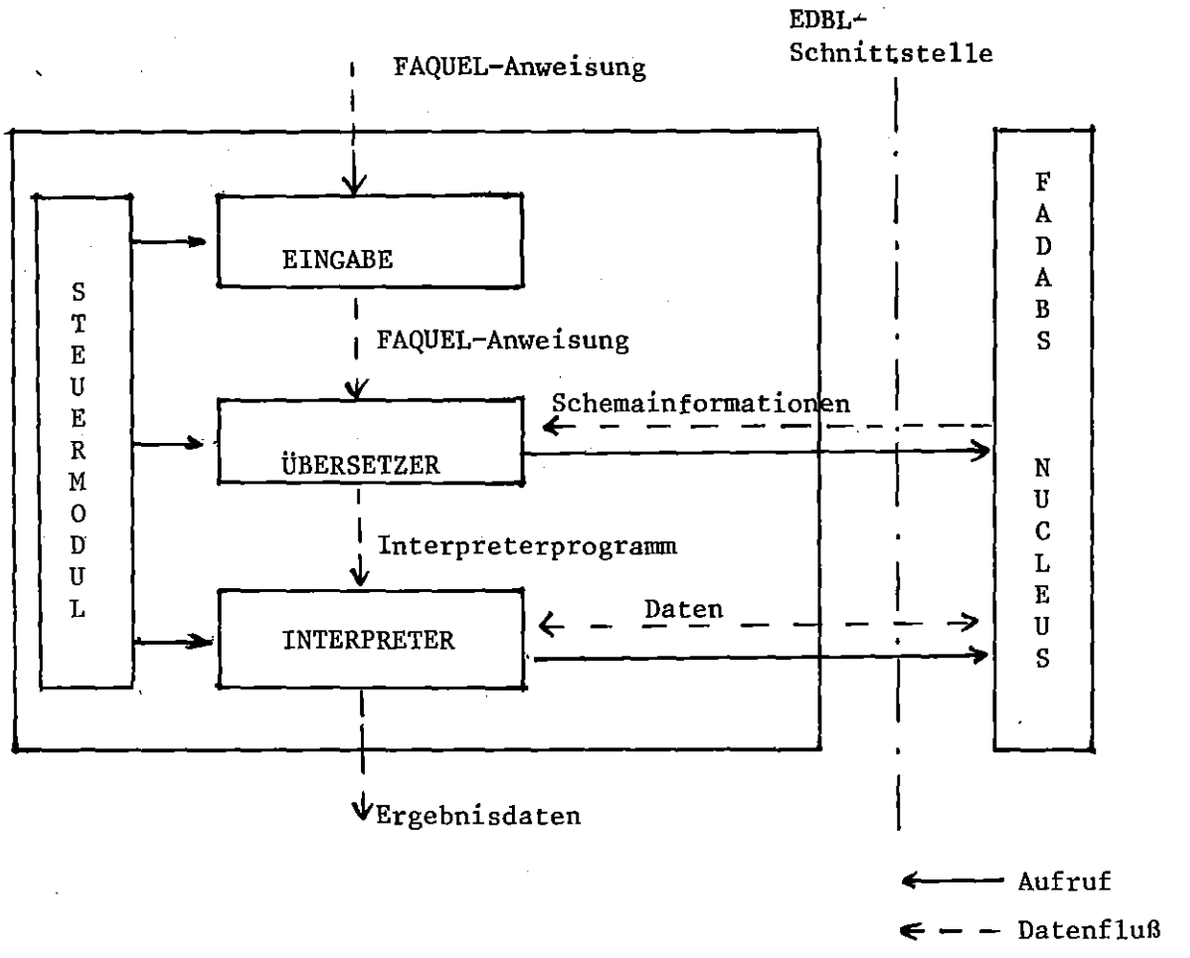


Bild 4: Systemstruktur

Bemerkungen:

Für den Scanner des Übersetzers und für den Interpreter konnte von vorhandenen, allgemein verwendbaren Moduln nach entsprechender Anpassung Gebrauch gemacht werden. Der Zerteiler des Übersetzers wurde mit Hilfe des zerteilererzeugenden Systems PGS (Parser Generating System) an der Universität Karlsruhe /De 77/ generiert. Das PGS benötigt als Eingabe die Syntax der Sprache in Form einer LALR(1)-Grammatik /AU 72/. Für FAQUEL wurde zunächst eine kontextfreie Grammatik /AU 72/ definiert und diese dann mit Unterstützung des PGS auf LALR(1)-Form gebracht.

10.2. Vorliegende Implementierung (Stand Dez. 1980)

Die vorliegende Implementierung realisiert im wesentlichen die Sprachelemente, die in den Abschnitten 3.1, 4.1, 4.2, 5.2.2, 5.2.3, 5.2.4, 5.2.5 (DELETE) beschrieben sind. Der mittels PGS erzeugte Zerteiler ist darüberhinaus in der Lage, die Syntaxanalyse für die Sprachelemente aus 3.2, 3.3, 3.4, 3.5, 3.6, 4.3, 5.2.5, 6, 8.1, 8.2, 8.3, 8.4, 8.5 durchzuführen.

Das Programm besteht aus ca. 8500 FORTRAN-Anweisungen einschließlich Kommentarzeilen. Der Hauptspeicherbedarf für die SIEMENS 330 beträgt ca. 47 K Wörter (1 Wort = 16 Bit) wenn das Programm linear gebunden ist, ca. 20 K Wörter segmentiert gebunden.

Referenzen

/ANS 77/ American National Standards Committee X3J3: draft proposed ANS FORTRAN. ACM SIGPLAN Notices, 11 (1976)

/AU 72/ Aho, A.V., Ullman, J.D.:
The Theory of Parsing, Translation, and Compiling.
Vol. 1/2. Prentice Hall Inc, 1972

/Ch 76/ Chamberlin, D.D., et al.:
SEQUEL2: A Unified Approach to Data Definition, Manipulation, and Control.
IBM J.Res. and Develop. 20, 6(Nov 1976), S. 560-574

/Co 70/ Codd, E.F.:
A Relational Model of Data for Large Shared Data Bases.
Comm. ACM 13 (1970), S. 377-387

/Co 71/ Codd, E.F.:
Relational Completeness of Data Base Sublanguages.
In: Rustin, R. (ed.): Data Base Systems.
Courant Computer Science Symposium 6, May 1971.
Prentice Hall Inc., 1971, S. 65-98

/De 77/ Dencker, P.:
unveröffentlichte Ergebnisse (1977)

/Gr 71/ Gries, D.:
Compiler Construction for Digital Computers.
John Wiley & Sons Inc, New York 1971

/JPT 80/ Jaeschke, A., Polster, F.J., Trauboth, H.:
A Generable Database System for the Use in Process Information Systems.
In: Proceedings COMSAC 80, Chicago, Oct 27-31, 1980.
IEEE Computer Society, S. 670-675

/Ke 79/ Kerpe, R.:
unveröffentlichter Bericht des KfK, 1979

/KP 79/ Kerpe, R., Polster, F.J.:
unveröffentlichter Bericht des KfK, 1979

/Lo 77/ Lough, D.E., Burns, A.D.:
An Analysis of Data Base Query Languages.
Masters's Thesis, Naval Postgraduate School, Monterey Cal., March 1977

/Mc 75/ McDonald, N., Stonebraker, M.:
CUPID, the friendly Query Language.
In: Proc. ACM Pacific Conf., San Francisco, April 1975

/Pe 75/ Pecherer, M.:
Efficient Retrieval in Relational Data Base Systems.
Memorandum No. ERL-M547, Electronics Research Laboratory,
College of Engineering, University of Calif., Berkeley, Okt. 1975

/Po78/ Polster, F.J.:
Ein Datenbanksystem für den Siemens Prozeßrechner 330.
In: Voges, U.: Tagungsbericht der 9. Jahrestagung des Siemens Prozeßrechner-Anwenderkreises I, Kernforschungszentrum Karlsruhe, KfK 2642, S. 227-244

/Po 80/ Polster, F.J.:
unveröffentlichter Bericht des KfK, 1980

/Re 77/ Reisner, P.:
Use of Psychological Experimentation as an Aid to Development of a Query Language.
IEEE Trans. Software Eng. SE-3, 3(1977)

/Sh 78/ Shneiderman, B.:
Improving the Human Factors Aspect of Database Interactions.
ACM Transactions on Data Base Systems, Vol. 3, No. 4 (Dec. 1978), S. 427-439

/SP 80/ Stöckle, D., Polster, F.J.:
Die Report-Definitionssprache RDL und ihre Implementierung im Report-Generator FAREG.
KfK-Bericht 2910, Kernforschungszentrum Karlsruhe, 1980

/Th 75/ Thomas, J.C., Gould, J.D.:
A Psychological Study of Query by Example.
In: Proc. AFIPS National Computer Conf., May 1975 Vol. 44, AFIPS Press, Montvale, N.J., 1975, S. 439-445

/Tr 79/ Tritschler, P.:
unveröffentlichte Ergebnisse (1977)

/Tr 80/ Tritschler, P.:
unveröffentlichter Bericht des KfK, 1980

/Va 77/ Vandijck, E.:
Towards a more Familiar Relational Retrieval Language.
Inform. Systems, 2 (1977), S. 159-169

/We 74/ Wedekind, H.:
Datenbanksysteme I.
Bibliographisches Institut, Mannheim 1974, Reihe Informatik/16

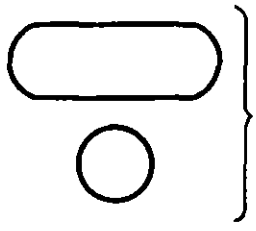
/Zl 75/ Zloof, M.M.:
Query by Example.
In: Proc. AFIPS National Computer Conf., May 1975 Vol. 44, AFIPS Press, Montvale, N.J., 1975, S. 431-438

Anhang 1:

FAQUEL-Syntax

Die Syntax der FAQUEL-Anweisungen wird in Form von Syntaxdiagrammen angegeben.

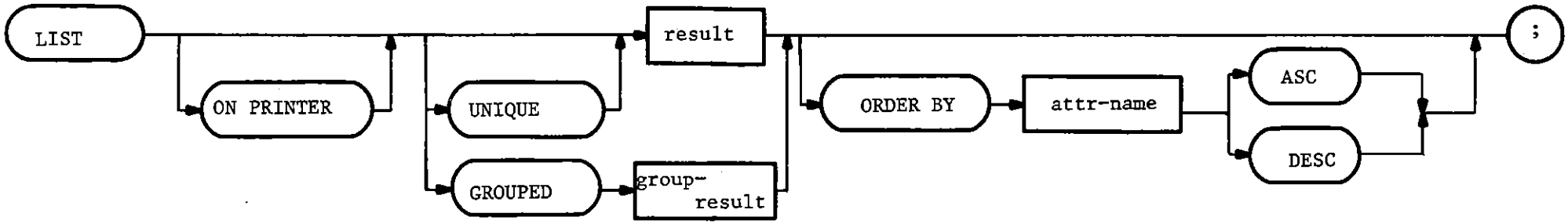
Zeichenerklärung



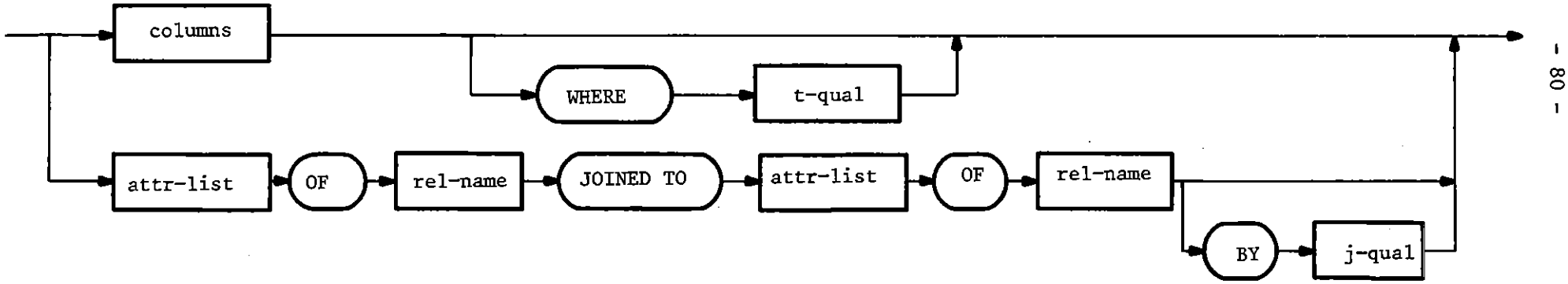
Terminalsymbole

Nichtterminalsymbole

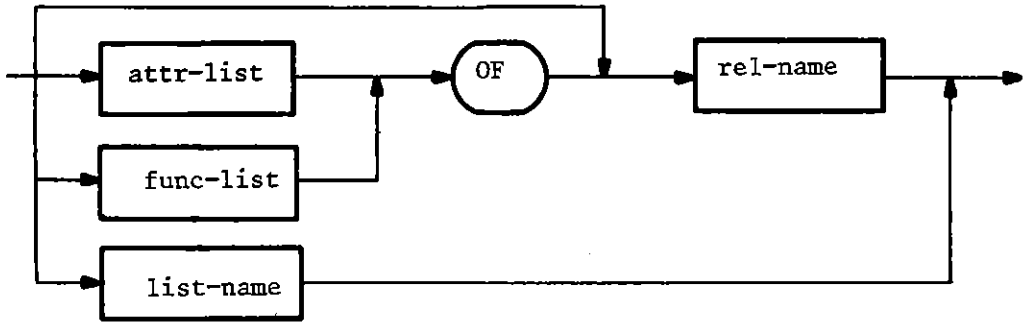
statement



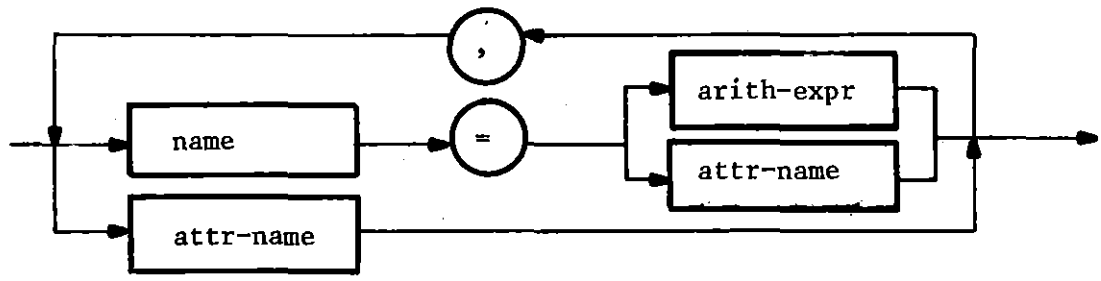
result



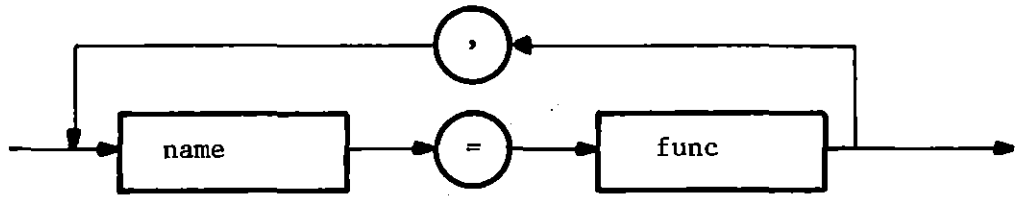
columns



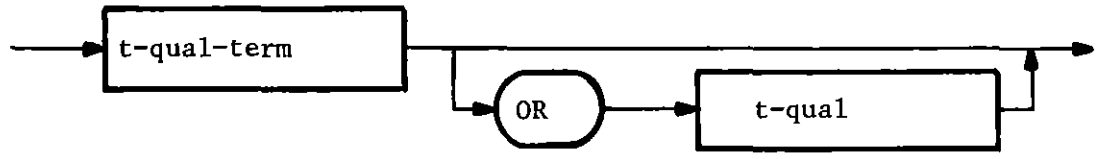
attr-list



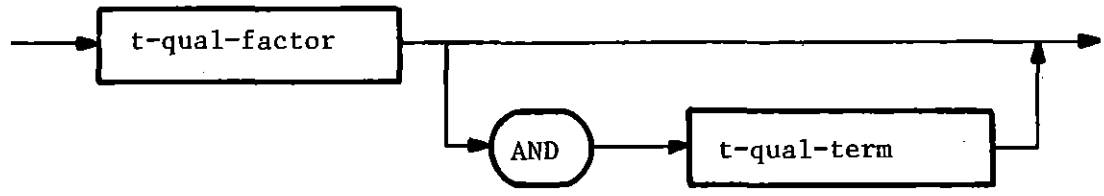
func-list



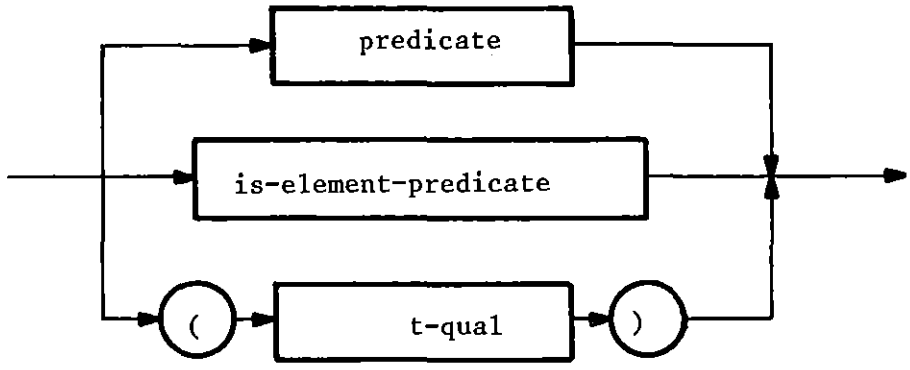
t-qual



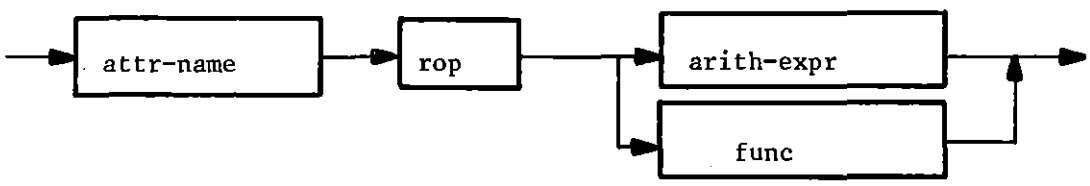
t-qual-term



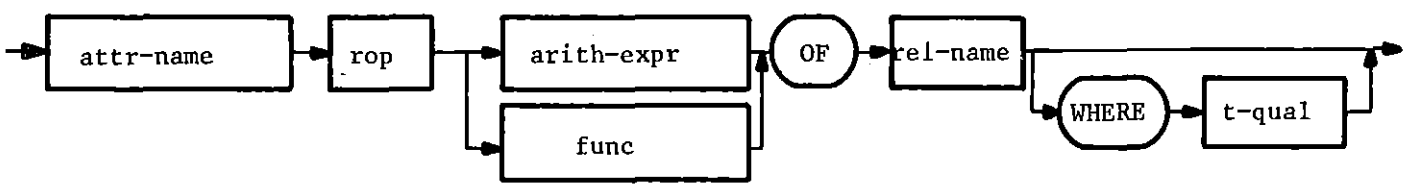
t-qual-factor



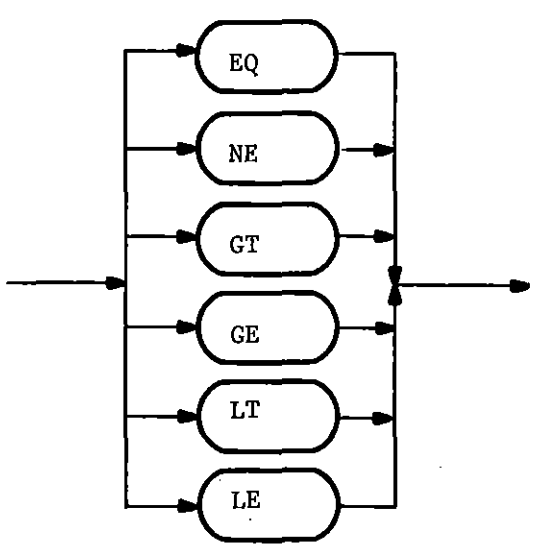
predicate



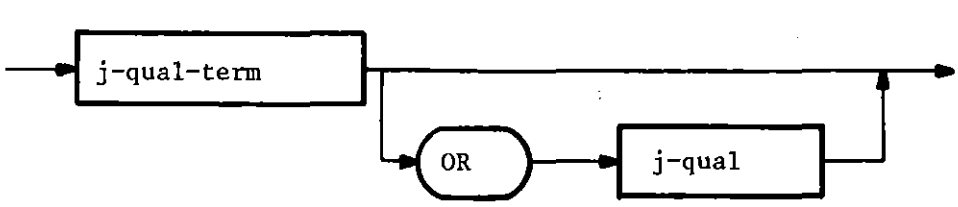
is-element-predicate



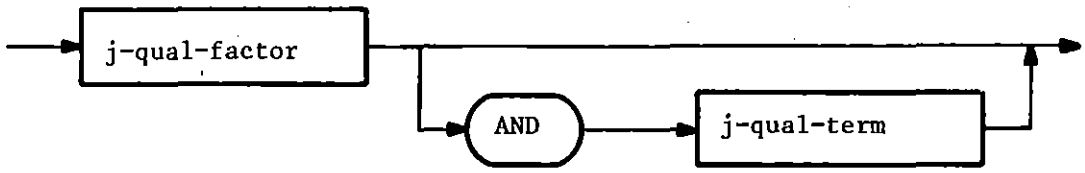
rop



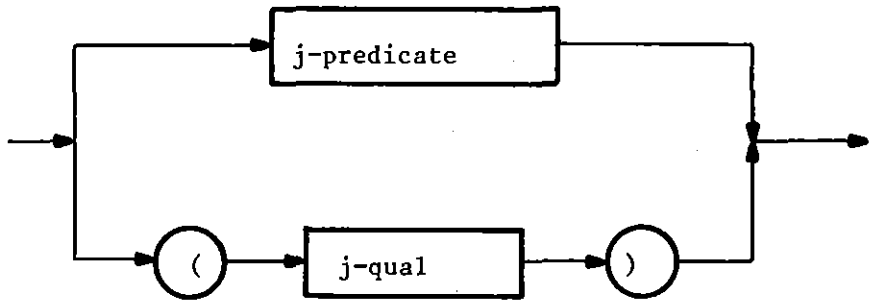
j-qual



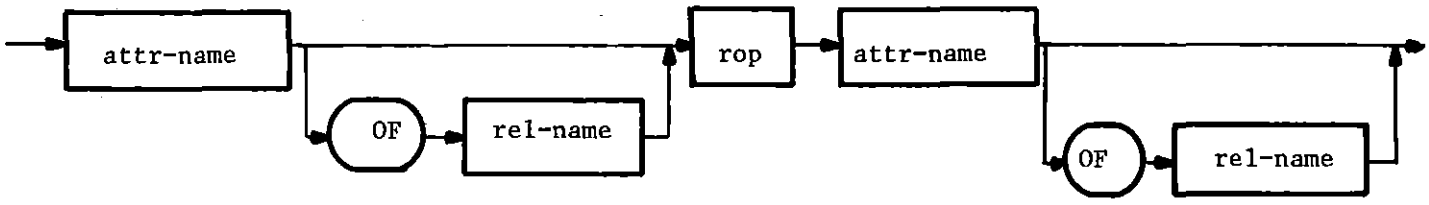
j-qual-term



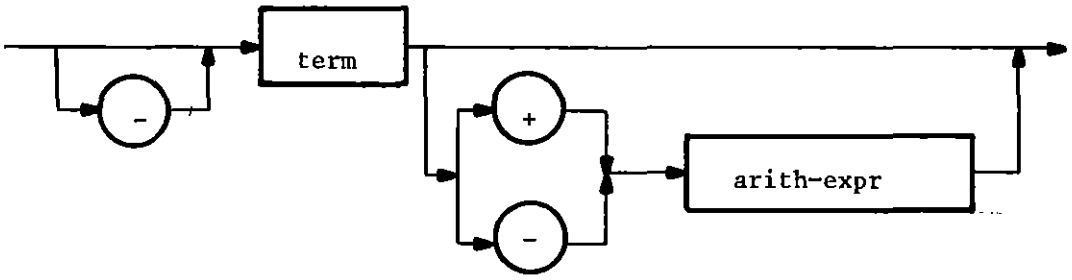
j-qual-factor



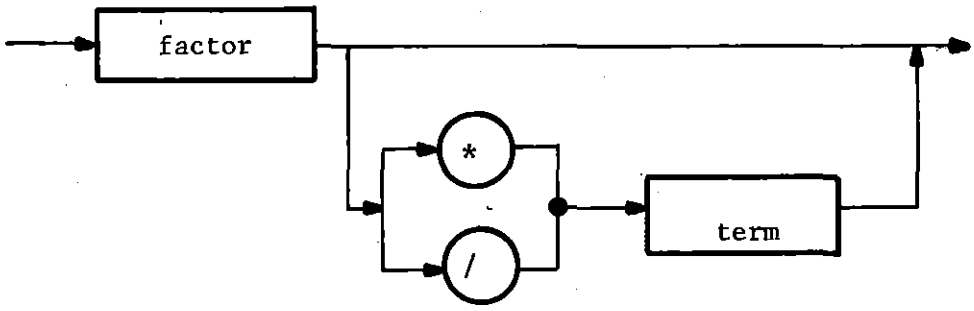
j-predicate



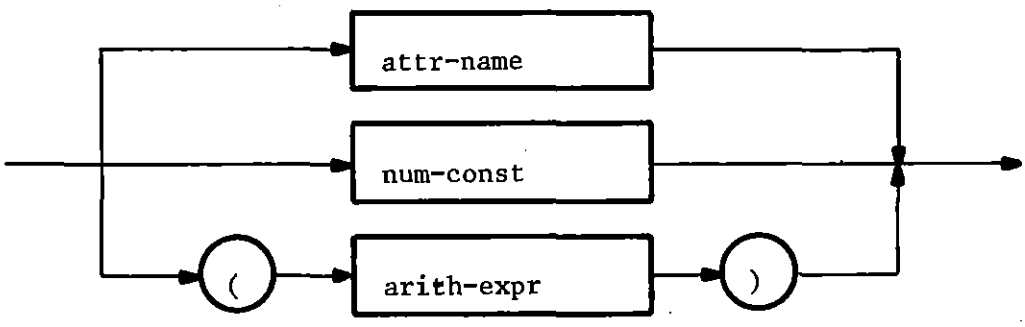
arith-expr



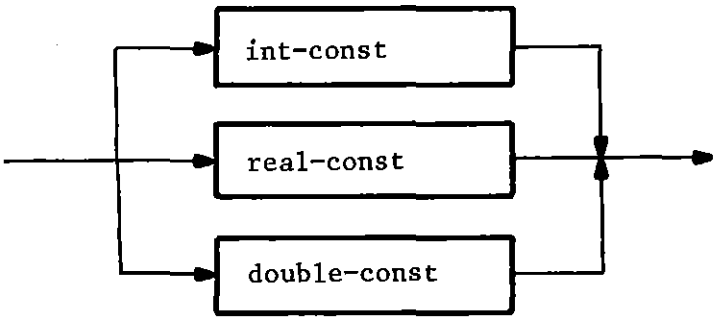
term



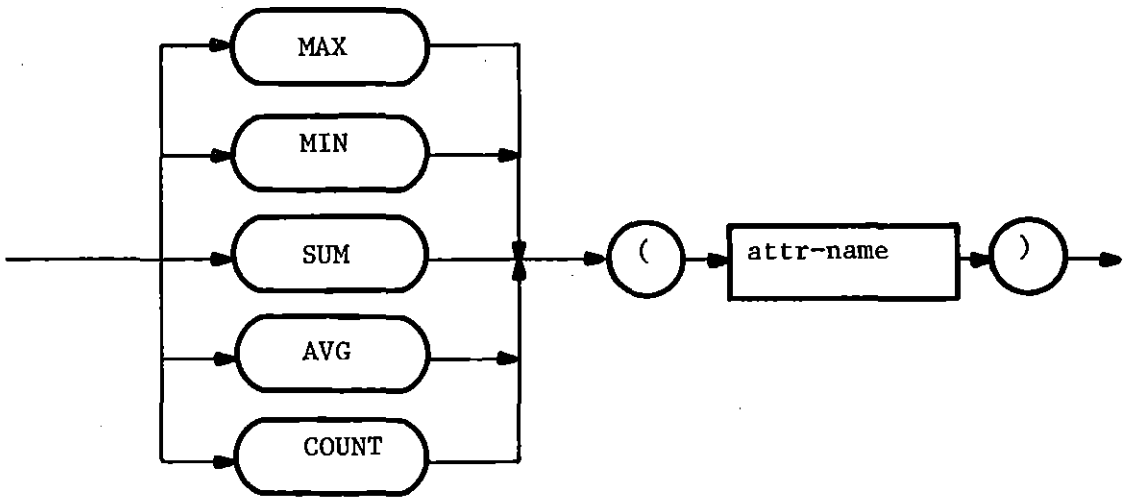
factor

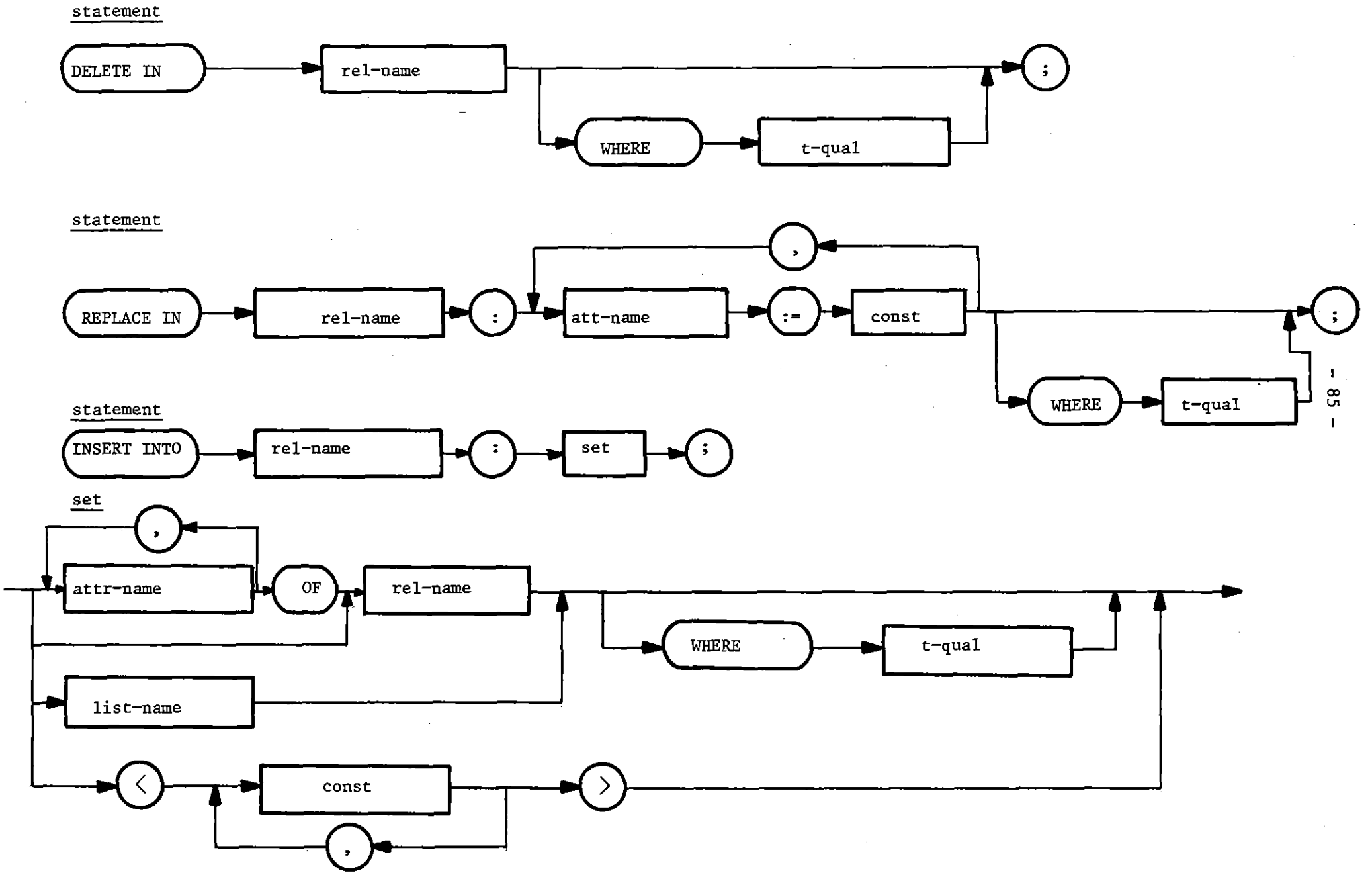


num-const

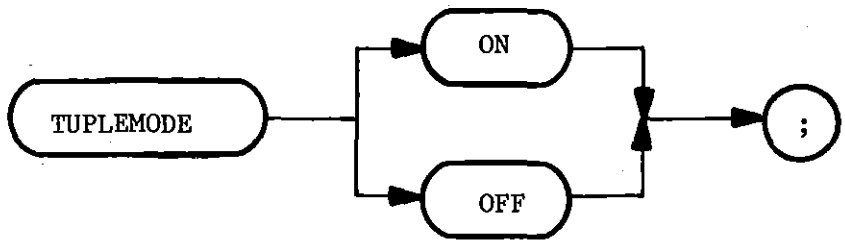


func

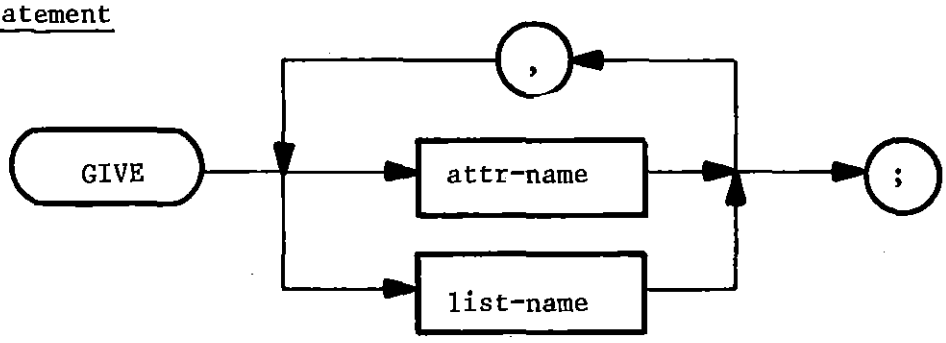



statement
DELETE IN
rel-name
;
WHERE
t-qual
statement
REPLACE IN
rel-name
:
,
att-name
:=
const
;
WHERE
t-qual
statement
INSERT INTO
rel-name
:
set
;
set
,
attr-name
OF
rel-name
WHERE
t-qual
list-name
<
const
,
>

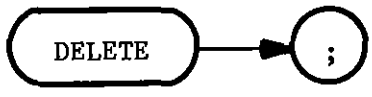
statement



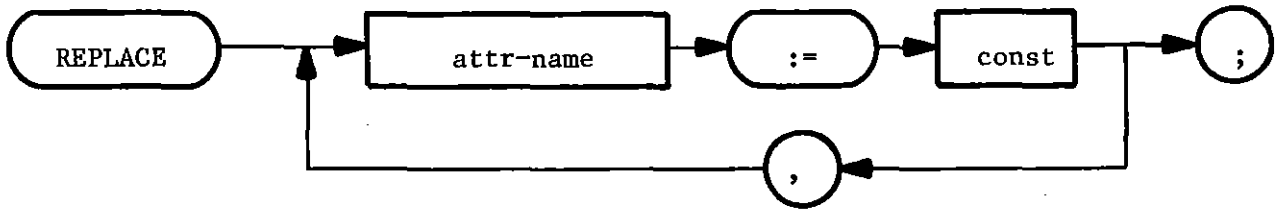
statement



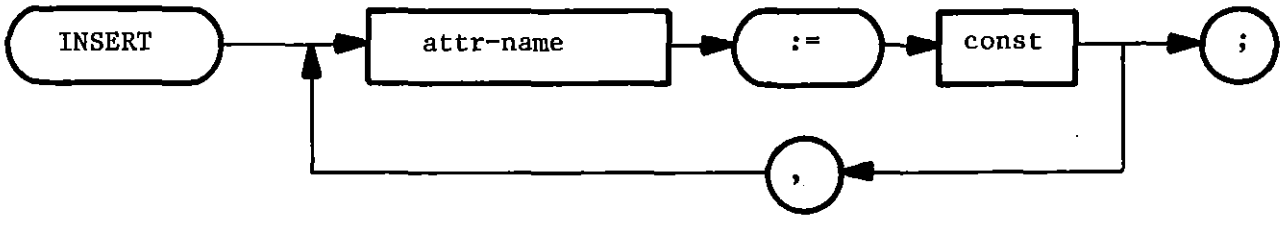
statement



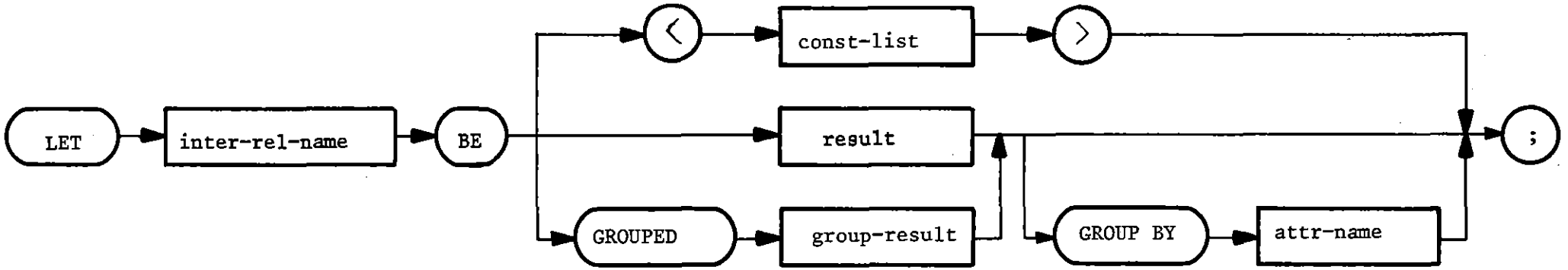
statement



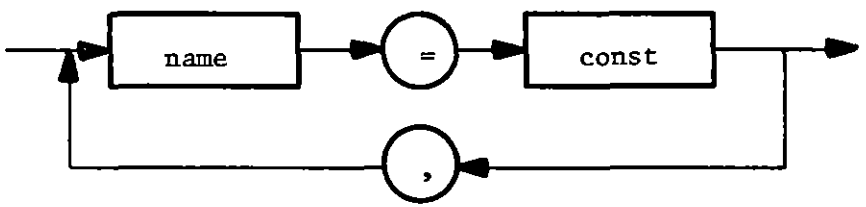
statement



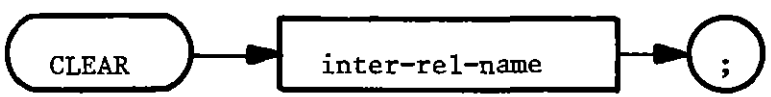
statement



const-list

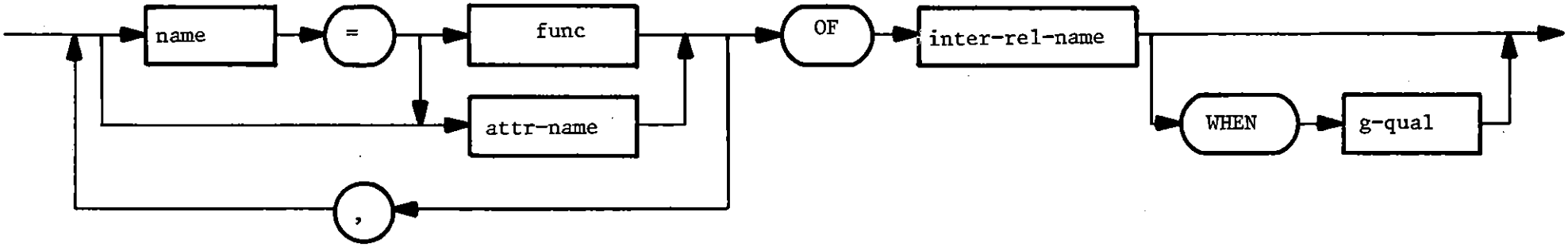


statement

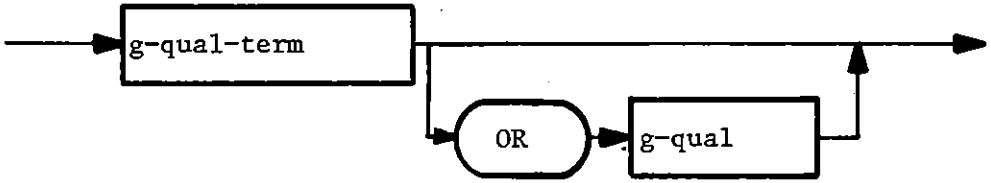


statement

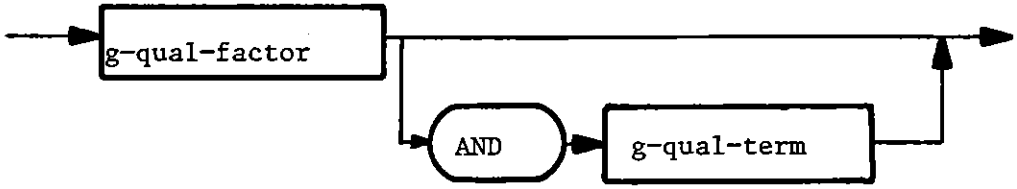
group-result



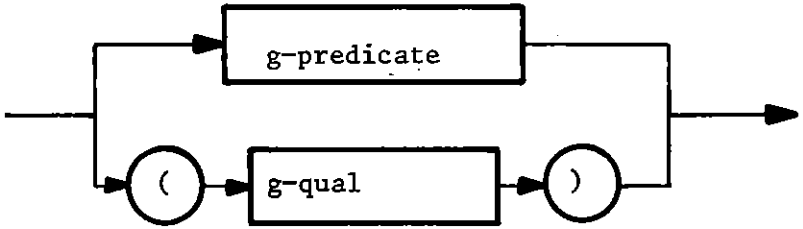
g-qual



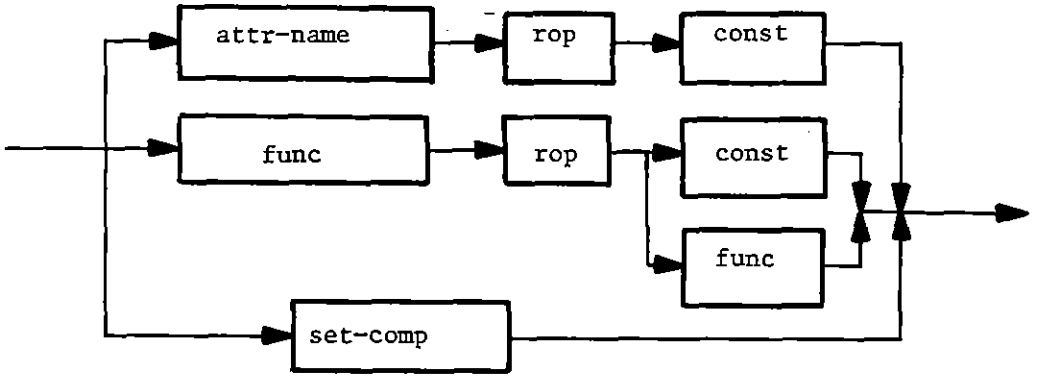
g-qual-term



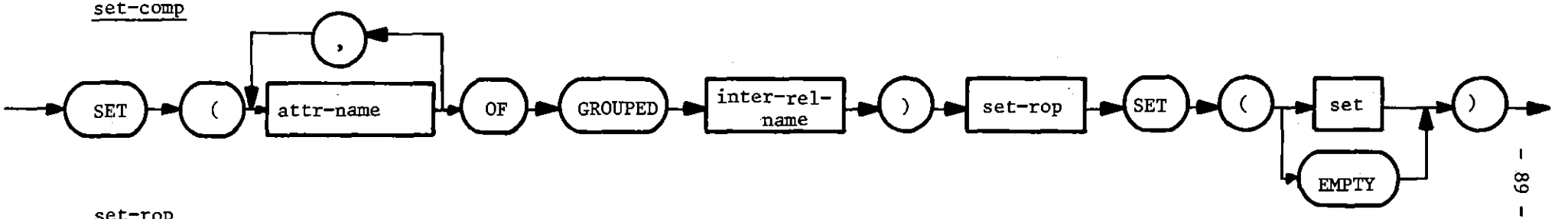
g-qual-factor



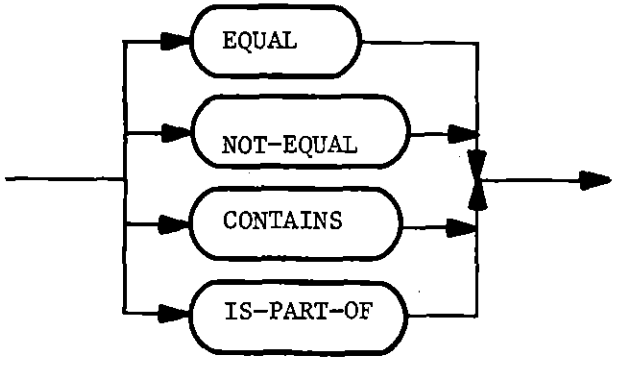
g-predicate



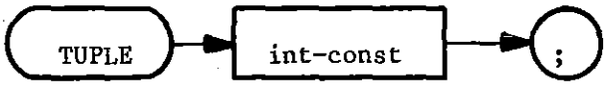
set-comp



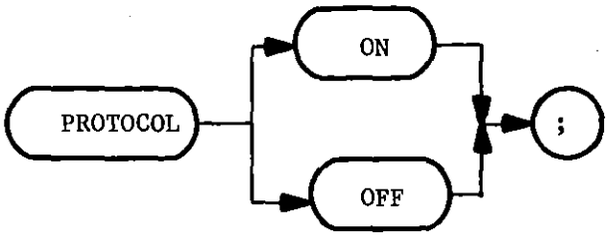
set-rop



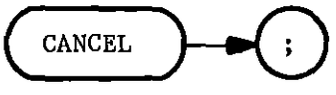
statement



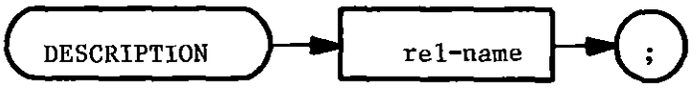
statement



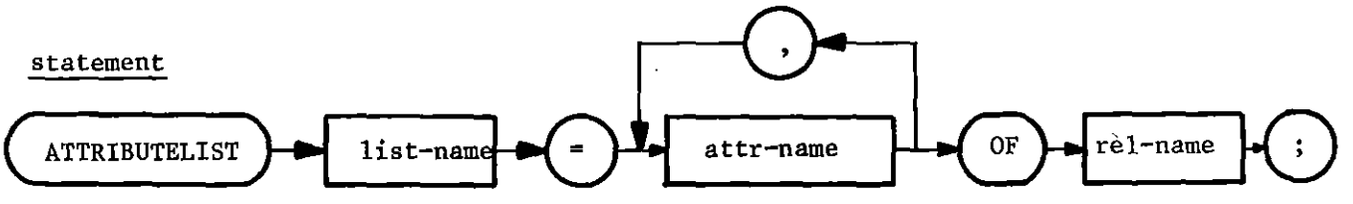
statement



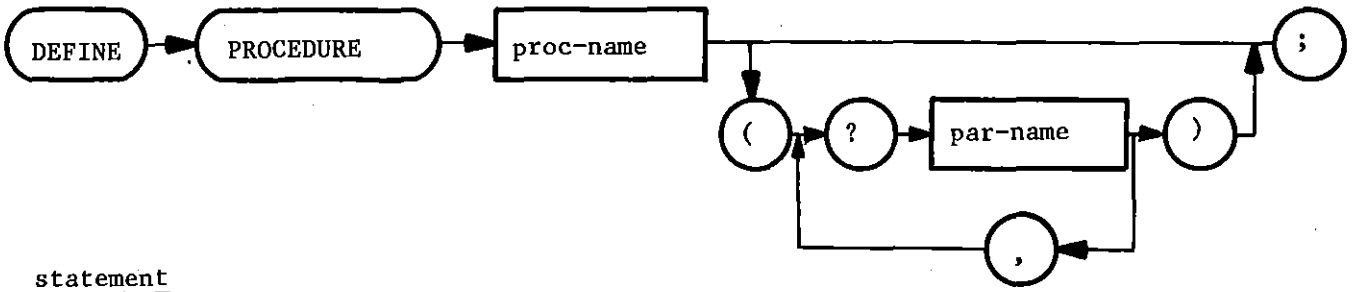
statement



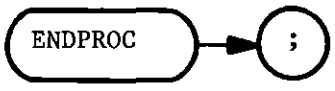
statement



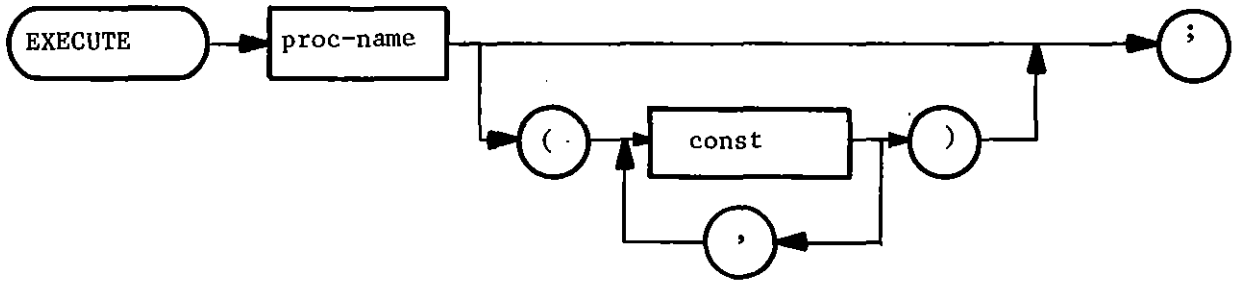
statement



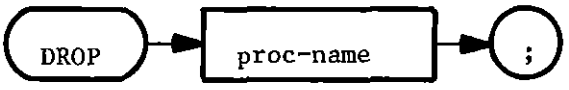
statement



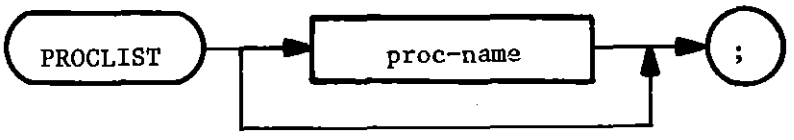
statement



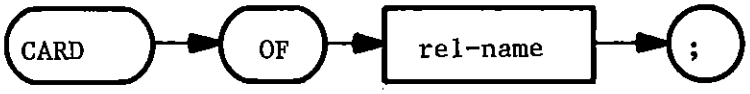
statement



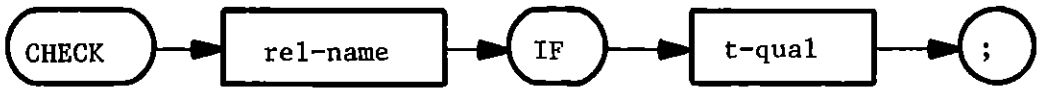
statement



statement



statement



Anhang 2:

Schlüsselwörter und Sonderzeichen in FAQUEL

Schlüsselwörter:

ASC ATTRIBUTELIST
BE BY
CANCEL CARD CHECK CLEAR
DEFINE DELETE DESC DESCRIPTION DROP
EMPTY ENDPROC EXECUTE
GIVE GROUP GROUPED
IF IN INSERT INTERMEDIATE INTO
JOINED
LET LIST
OF OFF ON ORDER
PRINTER PROCEDURE PROCLIST PROTOCOL
REPLACE
TO TUPLEMODE
UNIQUE
WHEN WHERE

Sonderzeichen:

+ - * / ( ) < > ; , : = ? :=