in: Datenbanksysteme in Büro, Technik und Wissenschaft, Proc. GI/SI-Fachtagung, Zürich, März 1989, Hrsg. T. Härder, Informatik-Fachberichte 204, Springer-Verlag, 1989, S. 389-407

Rekursion im MAD-Modell: Rekursivmoleküle als Objekte des Datenmodells

Harald Schöning
Universität Kaiserslautern

Überblick

Das Molekül-Atom-Datenmodell unterstützt die Datenbank-Verwaltung komplexer Objekte. Es bietet Operationen zum Zugriff, Ändern, Löschen und Einfügen an, die auf Mengen von dynamisch definierten Objekten arbeiten. Diese Objektdefinitionen können direkte und indirekte Rekursion enthalten, so daß die Berechnung der transitiven Hülle möglich ist. Im Gegensatz zu vielen anderen Vorschlägen für die Integration von Rekursion in Datenbanksysteme ist diese Hülle mit ihrer Struktur ein Objekt des Datenmodells, kann also mit dessen Operationen weiterverarbeitet werden. Eine Auswahl von Aggregations- und Verkettungsoperatoren ermöglicht die Lösung von Pfadproblemen durch Berechnung der generalisierten transitiven Hülle.

Abstract

The molecule atom data-model (MAD model) supports the management of complex objects in a database. There are operations to retrieve, update, delete, and insert sets of dynamically defined objects. The object definitions may contain direct and indirect recursion. Thus, transitive closure computations are possible. In contrast to many proposals for the integration of recursion in a database system published earlier, the transitive closure together with its structure can be represented within the data model. Path problems can be solved by a choice of various aggregation and concatenation operators, which may be applied to the generalized transitive closure.

1. Einleitung

Schon seit einigen Jahren wird der Einsatz von Datenbanksystemen nicht nur für die bekannten “klassischen” Anwendungsgebiete, sondern auch für die Datenverwaltung in Expertensystemen / vgl. Re87/, CAD-Systemen und ähnlichen “Non-Standard-Anwendungen” erwogen /BTW85, BTW87/. Für diese Anwendungsgebiete sind jedoch die herkömmlichen Datenbanksysteme ungeeignet /HR85/. Daher werden weltweit sogenannte Non-Standard-Datenbanksysteme entwickelt, von denen man sich eine bessere Unterstützung neuer Anwendungen verspricht. Neben neuen Verarbeitungskonzepten basieren diese auf Datenmodellen, die sich durch Eigenschaften wie

- Integration von Zeit und Unterstützung von Versionen und Alternativen,
- Fähigkeit zur Behandlung von Rekursion und
- Abbildung von “mehr” Semantik im Datenmodell, z.B. durch die Fähigkeit zur Darstellung ko-

mplexer (strukturierter) Objekte und zur Definition komplexerer Integritätsbedingungen

auszeichnen. Diese Datenmodelle sind entweder Erweiterungen bekannter Datenmodelle (meist des Relationenmodells) oder Neuentwicklungen. Im folgenden soll die Behandlung von Rekursion näher betrachtet werden. Dabei geht es nicht um Auswertungsaspekte, sondern um die Einbettung der Rekursion in Datenmodell und Anfragesprache. Als Motivation werden zunächst einige Beispiele für Rekursion gezeigt, die in den obengenannten Non-Standard-Anwendungen vorkommen.

2. Anwendungsbeispiele

Deduktive Datenbanken

Deduktive Datenbanksysteme sollen die Datenhaltung z.B. in Expertensystemen verbessern, die im allgemeinen weder die Nutzung von Sekundärspeicher noch Mehrbenutzerbetrieb unterstützt. Solche Datenbanksysteme besitzen die Fähigkeit zur Herleitung einer sogenannten intentionalen Datenbank. Aus einer “extensionalen” Datenbank (also aus explizit gespeicherten Daten) wird die intentionale Datenbank bei Bedarf durch einen Inferenzmechanismus rekursiv gewonnen. Eine einfache extensionale Datenbank könnte z.B. aus der Relation PERSON (Name, Elternteil) bestehen. Die intentionale Datenbank beinhaltet dann z.B. die Relation AHNEN (Name, Vorfahr), deren Daten aber nicht explizit abgespeichert sind, sondern bei Bedarf abgeleitet werden.

VLSI-Entwurf

Die Verwendung von Rekursion bietet sich natürlicherweise überall dort an, wo eine stufenweise Zerlegung oder Verfeinerung auftritt, besonders, wenn die Anzahl der Zerlegungsstufen nicht von vorneherein festliegt. Ein Beispiel hierfür ist der VLSI-Entwurf, bei dem ein bestimmter Chip aus Zellen bestimmter Funktionalität besteht, die wiederum aus anderen (ggf. Standard-) Zellen zusammengesetzt werden können, usw.

Beim VLSI-Entwurf bestimmt ein sogenannter Floorplan die geometrische Lage von Zellen. Dieser Floorplan kann durch einen sogenannten Slicing-Baum dargestellt werden (vgl. /Hä85/), der eine rekursive Flächenaufteilung beschreibt, und somit ein weiteres Anwendungsgebiet von Rekursion darstellt.

Versions- und Zeitverwaltung

Eine wichtige Forderung von Entwurfsanwendungen (CAD, VLSI-Entwurf) an Datenbanksysteme ist die Unterstützung der Darstellung eines Objektes in verschiedenen Beschreibungen. Dabei kann zwischen Versionen, Alternativen, Repräsentationen /Mi85/ unterschieden werden. Verschiedene Darstellungen stehen miteinander in Beziehung (z.B. “ist hervorgegangen aus” bei Versionen) und bilden zusammen ein Entwurfsobjekt. Da hierarchische Beziehungen zwischen den Darstellungen in nicht vorhersagbarer Tiefe vorliegen können, ist auch hier eine rekursive Vorgehensweise erforderlich.

3. Rekursion in Datenbanksystemen

Es gibt einige Vorschläge für die Integration einer stark eingeschränkten Rekursion in Datenbanksysteme, z.B. kennt QBE nach /DS86/ Rekursion nur bei binären Relationen, die einen Baum bilden. POSTGRES /SR86/ bietet die Möglichkeit, die transitive Hülle zu berechnen.

Eine Vielzahl von Aufsätzen (einige neuere sind: /HN84/, /Io86/, /De87/, /LMR87/) behandelt die effiziente Berechnung rekursiver Anfragen, ohne jedoch Aussagen über die Einbettung der Rekursion in das Datenmodell zu machen. /RS86/ und /Ul85/ fordern die Erweiterung von Datenbanksprachen um Horn-Klauseln, womit Rekursion formulierbar würde. Eine solche Vermischung herkömmlicher Datenbanksprachen mit völlig anders gearteten Konstrukten scheint aber zu inhomogenen Sprachansätzen zu führen, die die natürliche Integration der Rekursion im Sinne einer “einheitlichen”, homogenen Sicht der Datenbank nicht unterstützen.

Wie in /DS86/ festgestellt wird, ist es nicht sinnvoll, Rekursion in ihrer vollen Allgemeinheit in Datenmodelle einzubetten, da die Anfragesprache dann nicht mehr entscheidbar wäre. Als eine Rekursionsklasse mit “gutem Verhalten” wird dort die Klasse der Pfadprobleme weiter betrachtet. Das einfachste Pfadproblem ist die Berechnung der transitiven Hülle (also aller erreichbaren Knoten in einem Graphen). Zur Lösung komplexerer Pfadprobleme wird die Berechnung des sogenannten verallgemeinerten Hüllenoperators (“generalisierte transitive Hülle”) benutzt. Das bedeutet, daß neben der Ermittlung aller erreichbaren Knoten eines Graphen auch Informationen entlang der dabei verfolgten Wege (Pfade) berechnet werden. Dabei werden zwei Operatoren benötigt:

- der *Verkettungsoperator* berechnet aus den Informationen über einzelne Kanten Werte innerhalb eines Pfades.
- Durch den *Aggregationsoperator* werden diese zwischen den Pfaden zusammengefaßt.

Tabelle 1 zeigt die Wahl beider Operatoren für typische Pfadprobleme.

Problemtyp	Gesuchte Eigenschaft	Aggregations-operator	Verkettungs-operator
Kürzester Pfad	Länge oder Dauer	Minimum	+
Kritischer Pfad	Länge oder Dauer	Maximum	+
Breitester Pfad	Kapazität	Maximum	Minimum
Zuverlässigster Pfad	Verfügbarkeit	Maximum	*
Stückliste	Häufigkeiten	+	*
Alle Wege	Teilstrecken	$\cup$	Aneinanderfügen
Irgendein Weg	Teilstrecken	irgendeiner	Aneinanderfügen
Logische Deduktion	Prädikate	irgendeiner	Aneinanderfügen

Tabelle 1: Varianten des verallgemeinerten Hüllenoperators /Re87/, /DS86/

In /DS86/ wird die Berechnung des verallgemeinerten Hüllenoperators als vorläufiges Ziel der Integration von Rekursion in Datenbanksysteme gesehen. Drei Arten der Formulierung von Anfragen werden untersucht:

- **prozedural**: Solch eine auswertungsorientierte Rekursionsspezifikation durch den Benutzer

führt u.a. zu schlechter Optimierbarkeit.

- **logik-orientiert**: Eine Formulierung mittels Horn-Klauseln führt - in geringerem Maße - zu demselben Problem.
- **schablonenartig-deskriptiv**: Da die Rekursionsspezifikation nur über Aggregations- und Verkettungsoperator erfolgt, steht dem System ein großes Optimierungspotential offen.

Datenmodelle, die keine strukturierten Objekte kennen (wie etwa das Relationenmodell), können eine transitive Hülle nur über ihre Knoten, nicht aber in ihrem Zusammenhang darstellen, so daß die Information über die Struktur der transitiven Hülle nicht repräsentiert wird (“flache” Darstellung). NF^2 /Da86, SS86/ kennt statisch strukturierte Objekte (nicht-normalisierte Relationen mit fester Struktur). Da die Tiefe einer transitiven Hülle zum Anfragezeitpunkt nicht feststeht, kann sie nicht als strukturiertes Objekt des Datenmodells dargestellt werden /Li87/. Das in /KCB87/ vorgestellte Datenmodell erlaubt direkte Rekursion in sogenannten Konfigurationen, die aus einer Hierarchie von komplexen Objekten bestehen, so daß eine einfache transitive Hüllenberechnung möglich ist. Inwieweit die berechnete transitive Hülle im Datenmodell dargestellt und weiterverarbeitet werden kann, bleibt allerdings unklar.

Das Molekül-Atom-Datenmodell unterstützt dynamische strukturierte Objekte. Es erlaubt eine “strukturierte” Darstellung mit unbestimmter Tiefe, also auch die Darstellung der transitiven Hülle als Objekt des Datenmodells. Zusammen mit den angebotenen Operationen zur Berechnung der transitiven Hülle und der generalisierten transitiven Hülle stellt es ein mächtiges Werkzeug zur Behandlung von Rekursion dar.

Im folgenden werden zunächst einige grundlegende Eigenschaften dieses Modells vorgestellt. Anschließend werden die Ausdrucksmöglichkeiten für Rekursion präsentiert und an Beispielen verdeutlicht.

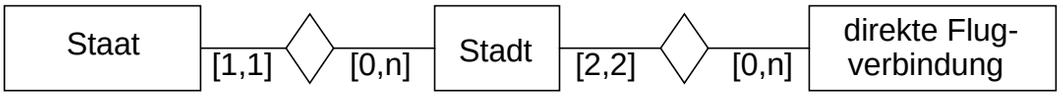
4. Einige Grundzüge des Molekül-Atom-Datenmodells

Das Molekül-Atom-Datenmodell (kurz MAD-Modell, /Mi88/) ist ein auf dem Relationenmodell aufbauendes Datenmodell zur Handhabung von komplexen Objekten, den sogenannten Molekülen. Diese werden zum Bearbeitungszeitpunkt dynamisch aus Basisobjekten, den sogenannten Atomen, aufgebaut. Atome entsprechen in etwa den Tupeln des Relationenmodells. Ein Atom gehört immer zu einem bestimmten Atomtyp (der einer Relation vergleichbar ist), und setzt sich aus Attributen verschiedener Datentypen zusammen.

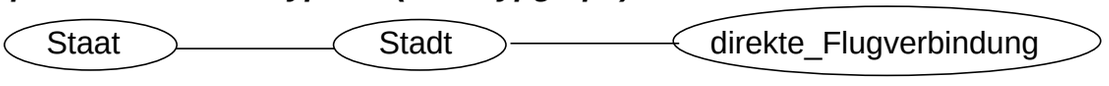
Neben den aus vielen herkömmlichen Datenmodellen bekannten Datentypen (wie INTEGER, REAL, etc.) sind für Attribute auch komplexere Datentypen wie TIME, LIST und ARRAY zugelassen. Jedes Atom besitzt genau ein Attribut vom Typ IDENTIFIER, das die eindeutige Identifikation des Atoms durch Verwendung eines Surrogatkonzeptes garantiert. 1:1, 1:n und n:m-Beziehungen zwischen Atomtypen lassen sich mittels Attributen des Typs REFERENCE darstellen: In jedem der beiden zu einer solchen Beziehung gehörenden Atomtypen gibt es ein REFERENCE-Attribut, das explizit an seinen Partner im anderen Atomtyp gebunden ist. Auf diese Weise entstehen soge-

nannte Atomtypnetze. Beispiel 1 zeigt ein Atomtypnetz für das angegebene Entity-Relationship-Diagramm: für jeden Entity-Typ wird ein Atomtyp definiert; die 1:n-Beziehung Staat-Stadt und die n:m-Beziehung Stadt — Flugverbindung werden jeweils durch ein Paar von REFERENCE-Attributen modelliert. In dem Atomtypgraphen wird diese Tatsache durch die ungerichtete Verbindung der Atomtypen dargestellt.

einfaches Entity-Relationship-Diagramm (direkte Linienflugverbindungen)



entsprechendes Atomtypnetz (Atomtypgraph)

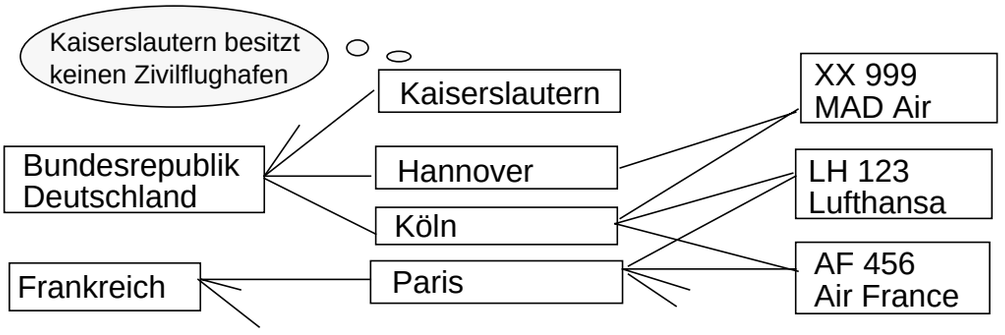


Beispiel 1: Umsetzung eines Entity-Relationship-Diagramms in ein Atomtypnetz

Der Wert eines REFERENCE-Attributes des einen Atomtyps ist eine Menge von IDENTIFIER-Werten der an der Beziehung beteiligten Atome des anderen Atomtyps, und umgekehrt. Dabei kann für die Kardinalität der Menge eine Ober- und eine Untergrenze angegeben werden. Somit könnte obiges Atomtypnetz durch folgende Atomtypdefinitionen beschrieben werden:

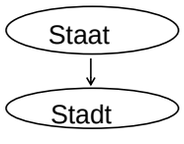
```
CREATE_ATOM_TYPE Stadt:
    (Stadt_Id : IDENTIFIER;
    Name : String;
    Nation : REFERENCE TO Staat.Städte (1,1)
    Flüge : REFERENCE TO direkte_Flugverbindung.zwischen (0,VAR);
    ...)


CREATE_ATOM_TYPE Staat:
    Staat_Id : IDENTIFIER;
    Name : String;
    Städte : REFERENCE TO Stadt.Nation (0, VAR);
    ...)

CREATE_ATOM_TYPE direkte_Flugverbindung:
    Flug_Id : IDENTIFIER;
    Flug_Nr : INTEGER;
    Linie : String;
    zwischen: REFERENCE TO Stadt.Flüge (2,2);
    ...)
```

Die Werte der REFERENCE-Attribute spezifizieren eine Netzstruktur auf Atomen, die bidirektional ist, da es zu jeder Referenz eine sogenannte Gegenreferenz gibt (Aus dem oben Gesagten folgt nämlich: Wenn es ein REFERENCE-Attribut des Atoms a1 den Wert {b1,b2} hat, dann gibt es in b1 und in b2 jeweils ein REFERENCE-Attribut, das den Wert a1 enthält). Beispiel 2 zeigt ein Atomnetz zu dem Atomtypnetz aus Beispiel 1. IDENTIFIER-Attribute sind nicht dargestellt. Die Werte von REFERENCE-Attributen sind durch die Verbindungslinien zwischen den Atomen symbolisiert. Jede ungerichtete Verbindung entspricht einem Referenz-/Gegenreferenz-Paar.

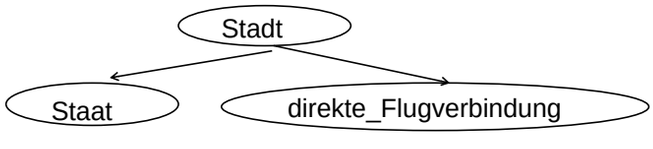


Beispiel 2: Ein Atomnetz

Die Datenmanipulationssprache des MAD-Modells bietet nun die Möglichkeit, in Manipulations- und Retrieval-Anweisungen aufbauend auf dem jeweiligen MAD-Schema dynamisch komplexe Objekttypen (Molekültypen) zu definieren. Dazu werden Knoten des Atomtypnetzes ausgewählt und Kanten zwischen diesen mit einer Richtung versehen, so daß ein zusammenhängender gerichteter Graph (Molekültypgraph) entsteht. Von genau einem Atomtyp dieses Graphen, dem sogenannten Wurzelatomtyp, müssen allen anderen erreichbar sein. Da die über REFERENCE-Attribute ausgedrückten bidirektionalen Beziehungen symmetrisch verwendbar sind, wird die Richtung der Beziehungen zwischen Atomtypen erst zu diesem Zeitpunkt festgelegt. Die Notation zur Definition eines Molekültyps soll in etwa die Graphstruktur widerspiegeln. Ausgehend vom Wurzelatomtyp werden die zu verwendenden REFERENCE-Attribute und die abhängigen Atomtypen angegeben. Eine (gerichtete) Beziehung wird durch "-" symbolisiert. So entspricht z.B. **Staat.Städte - Stadt** dem Molekültypgraphen:



Der Attributname des zu verwendenden REFERENCE-Attributes kann entfallen, wenn die Beziehung eindeutig ist, wenn es also zwischen den beiden Atomtypen nur eine Beziehung gibt. Im obigen Fall wäre daher **Staat - Stadt** auch erlaubt. Verzweigungen im Molekülgraphen werden durch entsprechende Klammerung ausgedrückt. **Stadt - (.Nation - Staat, .Flüge - direkte_Flugverbindung)** beschreibt folgenden Molekültypgraphen:



Auch hier ist die Kurzfassung zulässig: **Stadt - (Staat, direkte_Flugverbindung)**

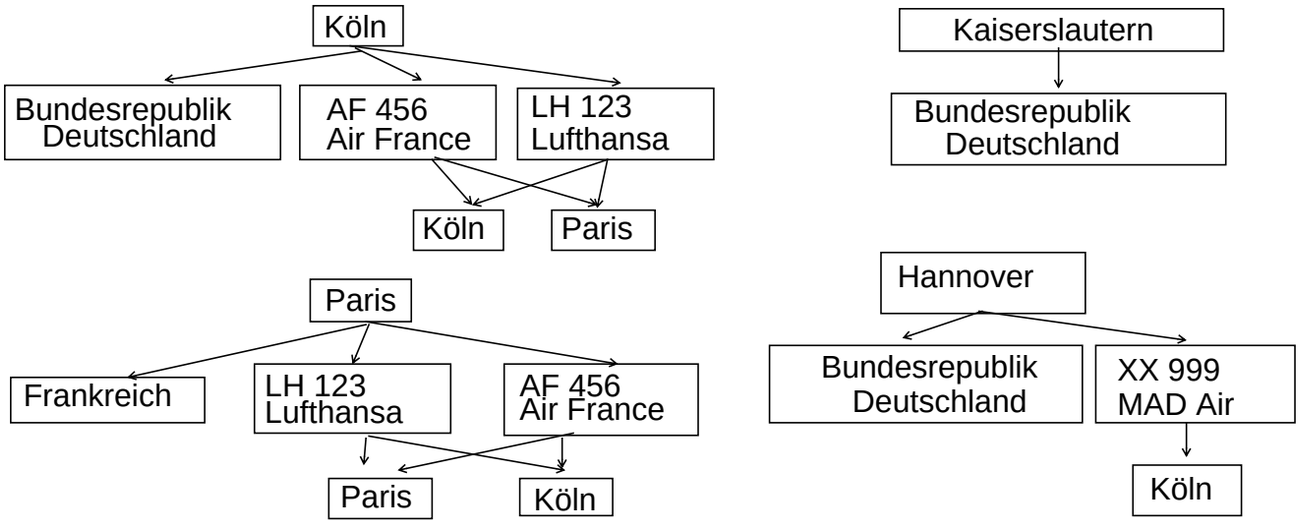
Da der Wurzelatomtyp erst durch die Definition des Molekültyps festgelegt wird, ist natürlich auch die Molekültypdefinition **direkte_Flugverbindung - Stadt -Staat** erlaubt. Die mehrfache Aufnahme eines Atomtyps in einen Molekültyp ist möglich. Jedes Auftreten desselben Atomtyps muß dann zur Identifikation einen sogenannten Rollennamen erhalten. Sollen z.B. Start- und Zielflughafen in einem Molekültyp differenziert werden, obwohl es im Atomtyp direkte_Flugverbindung eine solche Unterscheidung nicht gibt, kann dies durch Vergabe von Rollennamen geschehen: **Start (Stadt) - direkte_Flugverbindung - Ziel (Stadt).**

Die Moleküle, die zu einem Molekültyp gehören, lassen sich folgendermaßen beschreiben:

- Für jedes Atom des Wurzelatomtyps existiert genau ein solches Molekül.
- Zu einem Molekül gehören alle Atome, die über eine in der Molekültypdefinition angegebene Beziehung direkt oder transitiv mit dem Wurzelatom verbunden sind.
- Ein Molekül entspricht somit einem gerichteten zusammenhängenden Graphen über dem Atomnetz.

Moleküle bestehen aus Atomen mit deren Attributen und sogenannter Strukturinformation, die die Molekülstruktur beschreibt und aus den REFERENCE-Attributen der Atome abgeleitet wird. Bei der graphischen Darstellung von Molekülen wird die Strukturinfomation mit Hilfe von Pfeilen ausgedrückt.

Die Definition **Start (Stadt) - (Staat, direkte_Flugverbindung - Ziel(Stadt))** bildet aus dem Atomnetz von Beispiel 2 folgende vier Moleküle:



In der Rolle **Ziel** sind aufgrund der Molekültypdefinition beide Endpunkte jeder Flugverbindung enthalten. Daher kommt es, daß Köln bzw. Paris zweimal in demselben Molekül enthalten sind.

Als Beispiel für die Anwendung von Molekültypdefinitionen soll im folgenden die Retrieval-Anweisung SELECT näher beschrieben werden. Um die Lesbarkeit zu gewährleisten, werden die Molekültypgraphen zu den einzelnen Beispielen nicht im Text, sondern im Anhang angegeben. Eine SELECT-Anweisung besteht aus einer Projektionsklausel, einer Definitionsklausel und einer Restriktionsklausel und beschreibt eine (dynamisch zu erzeugende) Menge von Molekülen:

SELECT <Projektionsklausel>
FROM <Definitionsklausel>
WHERE <Restriktionsklausel>

Die **Definitionsklausel** spezifiziert den Molekültyp, auf den sich die Anfrage bezieht. Es können auch mehrere Molekültypen spezifiziert sein, wodurch ein kartesisches Produkt der Molekültypen gebildet wird. Nur solche Moleküle gehören zum Ergebnis, die den in der **Restriktionsklausel** angegebenen Ausdruck erfüllen. Jeder nicht-quantifizierte Term dieses Ausdrucks wird als existentiell quantifiziert angesehen. Die **Projektionsklausel** spezifiziert die Teile der ermittelten Moleküle (also Attribute oder Atome), die zum Ergebnis gehören sollen. Das Schlüsselwort ALL steht für alle vorkommenden Atome und Attribute. Mit ALL_BUT können gezielt Atome oder Attribute ausge-

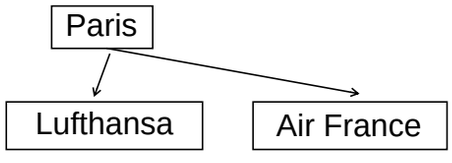
schlossen werden. Dabei muß allerdings immer ein zusammenhängendes Atomnetz bestehen bleiben, das das Wurzelatom enthält. In der Projektionsklausel können auch sogenannte virtuelle Attribute enthalten sein, deren Wert mittels einer Berechnungsvorschrift zu ermitteln ist.

Die Frage: *Von welchen französischen Städten aus und mit welchen Fluglinien kann man direkt nach Köln fliegen?* wird folgendermaßen formuliert:

```
SELECT Start (Name), direkte_Flugverbindung (Linie)
FROM   Start(Stadt)-(Staat, direkte_Flugverbindung-Ziel(Stadt))
WHERE  Staat.Name = 'Frankreich' AND Ziel.Name = 'Köln'
```

1

und liefert mit den Daten aus Beispiel 2 das Ergebnis:



Die Definitionsklausel legt also eine Menge von Molekülen fest, von denen die in der Projektionsklausel spezifizierten Teile ausgegeben werden, falls die Moleküle die Restriktionsklausel erfüllen. Die Molekülmenge kann nicht nur durch einen Typgraphen, sondern auch durch Angabe einer SELECT-Anweisung spezifiziert werden (Geschlossenheit des Sprachansatzes).

Die Konvertierungsfunktion **VALUE** erzeugt aus Molekülmengen, deren Moleküle aus genau einem Atom mit genau einem Attribut (vom Datentyp t) bestehen, eine Liste des Typs t. Soll aus Atomen *innerhalb eines Moleküls* eine Liste erzeugt werden, so steht hierfür die Funktion **MOLAGG** zur Verfügung.

Beispiel: *Welche Stadt hat überdurchschnittlich viele Flugverbindungen?*

```
SELECT ALL
FROM   Stadt
WHERE  COUNT (Flüge) > AVG (VALUE ( SELECT Anzahl:=COUNT (Flüge)
                                    FROM   Stadt))
```

Die Verwendung von SELECT-Anweisungen ist auch in der Projektionsklausel möglich. Dadurch können Teilmoleküle wertabhängig aus dem Ergebnismolekül ausgeblendet werden (qualifizierte Projektion). Das Schlüsselwort **RESULT** in der Definitionsklausel verdeutlicht, daß es sich dabei um eine Operation auf dem durch Definitions- und Restriktionsklausel der umgebenden Anfrage bestimmten Ergebnis handelt.

Beispiel: *Welche <u>anderen</u> Städte erreicht man von Köln direkt mit welchen Linien?*

```
SELECT Start(Name), direkte_Flugverbindung(Linie),
                    Ziele :=( SELECT Ziel (Name)
                              FROM   RESULT
                              WHERE  Ziel.Name <> 'Köln')
FROM   Start(Stadt)-direkte_Flugverbindung-Ziel(Stadt);
WHERE  Start.Name='Köln'
```

2

Da Köln, wie bereits gesehen, auch als Ziel-Stadt im Ergebnismolekül vorhanden ist, muß es explizit ausgeblendet werden. Die qualifizierte Projektion darf die Zusammenhangseigenschaft des Ergebnismoleküls nicht zerstören.

5. Rekursion im MAD-Modell

Nachdem nun einige Grundzüge des MAD-Modells vorgestellt wurden, soll die Handhabung von Rekursion näher betrachtet werden. In diesem Kapitel werden die wichtigsten Aspekte der Berechnung der transitiven Hülle vorgestellt, während das anschließende Kapitel dann die generalisierte transitive Hülle behandelt.

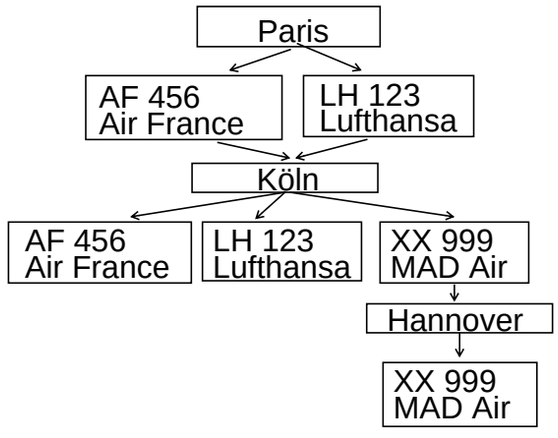
Die Integration von Rekursion in das MAD-Modell erlaubt es, einen Atomtyp nicht nur in einer festen Anzahl in verschiedenen Rollen in ein Molekül aufzunehmen, sondern “beliebig oft” entlang einer bestimmten Beziehung. So ist es in unserem Beispiel nicht nur interessant, welche Städte von einem bestimmten Startpunkt aus direkt erreicht werden können, sondern auch, welche Städte überhaupt (mit beliebig vielen Zwischenstops) zu erreichen sind. Es ist also nach der transitiven Hülle von direkte_Flugverbindung-Stadt gefragt. Ein entsprechender Molekültypgraph muß eine Schleife enthalten. Moleküle mit einem solchen Molekültypgraphen heißen Rekursivmoleküle. Rekursionsbildung ist im MAD-Modell dort möglich, wo Beziehungen zwischen Atomtypen definiert sind. Dies entspricht den in der Literatur (z.B. /DS86/, /Re87/) zur Rekursionsbildung herangezogenen Primärschlüssel-Fremdschlüssel-Beziehungen.

Ein Rekursivmolekül setzt sich aus einer unbestimmten Anzahl von typmäßig gleichen Komponentenmolekülen zusammen, die über eine ausgezeichnete Beziehung (die sogenannte rekursionsbildende Beziehung) miteinander verbunden sind. Der Typgraph eines Rekursivmolekültyps enthält also eine Schleife. Zu einem Rekursivmolekül gehört genau ein Komponentenmolekül in der Rekursionsstufe 0 (in Analogie zum Wurzelatom eines Moleküls). Zur Rekursionsstufe n+1 gehören alle Moleküle des Komponentenmolekültyps, auf deren Wurzelatom ein Wert der rekursionsbildenden REFERENCE-Attribute von Komponentenmolekülen in Stufe n verweist. Allerdings gehört ein Komponentenmolekül nur in der kleinstmöglichen Rekursionsstufe zum Rekursivmolekül, ist also nur einmal in jedem Rekursivmolekül enthalten. Durch diese Definition wird ein maximaler, gerichteter azyklischer Ausschnitt des Atomnetzes betrachtet (analog zu /Li87/ werden Schleifen vermieden). Da die Datenbank endlich ist, existiert ein endlicher Fixpunkt der Rekursion, so daß nur endliche Rekursivmoleküle entstehen können.

Beispiel: Berechnung der transitive Hülle **Stadt-direkte_Flugverbindung** für jede Stadt.

```
SELECT  ALL
FROM    Stadt-direkte_Flugverbindung
            RECURSIVE direkte_Flugverbindung.zwischen-Stadt
```

Ein Ergebnismolekül für das Atomnetz aus Beispiel 2. Die weiteren Ergebnismoleküle sind dem Anhang zu entnehmen.



anzusprechen durch:

Stadt(0)

direkte_Flugverbindung(0)

Stadt(1)

direkte_Flugverbindung(1)

Stadt(2)

direkte_Flugverbindung(2)

Teile von Rekursivmolekülen können folgendermaßen angesprochen werden:

- ALL_REC bezeichnet die Zusammenfassung aller Rekursionsstufen.
- (n) bezeichnet die Rekursionsstufe n. Das oberste Komponentenmolekül hat die Rekursionsstufe 0.
- (n..m) bezeichnet die Rekursionsstufen n bis m
- eine spezielle Aggregationsfunktion MAX_RECDEPTH gibt die Länge des längsten Pfades in dem Rekursivmolekül (also die größte Rekursionsstufe) an.
- LAST bezeichnet Blätter des Rekursivmolekülgraphen.

So könnte aus der oben berechneten transitiven Hülle die letzte Flugverbindung folgendermaßen entfernt werden:

SELECT ALL_BUT direkte_Flugverbindung(LAST)
FROM Stadt-direkte_Flugverbindung
RECURSIVE Flugverbindung.zwischen-Stadt

Die Molekülstruktur stellt die transitive Hülle “strukturiert” dar. Soll lediglich eine “flache” Sicht der erreichbaren Städte erzeugt werden, kann folgende Anfrage formuliert werden, die die Definition des virtuellen Attributes *erreicht* als Liste von Städtenamen enthält:

SELECT Stadt (0)(erreicht:=MOLAGG (Stadt (ALL_REC).Name))
FROM Stadt-direkte_Flugverbindung
RECURSIVE Flugverbindung.zwischen-Stadt

Die Rekursion muß nicht an der Wurzel des gesamten Moleküls beginnen; vielmehr kann ein beliebiger nicht-rekursiver Molekültyp als Wurzel des Ergebnisses spezifiziert werden, wie *Start* im folgenden Beispiel:

SELECT ALL
FROM Start(Stadt) - direkte_Flugverbindung - Stadt
RECURSIVE Stadt.Flüge-direkte_Flugverbindung

Oft soll nicht die gesamte transitive Hülle berechnet werden, sondern nur ein bestimmter Ausschnitt. Dabei sind mehrere Auswahlkriterien für diesen Ausschnitt denkbar:

- Die Rekursionstiefe soll ein bestimmtes Maximum nicht überschreiten.
- Sobald ein Rekursionspfad eine bestimmte Bedingung erfüllt, soll dessen Aufbau beendet werden. Diese Bedingung könnte zum Beispiel sein: “Die Stadt *Köln* ist im Pfad enthalten”.
- Nur solche Teile der transitiven Hülle sollen betrachtet werden, bei denen die rekursionsbildende Beziehung bestimmte Bedingungen erfüllt (z.B. Weiterflug nur mit einer Maschine derselben Linie).

Solche Auswahlkriterien können im MAD-Modell in der **UNTIL**-Klausel formuliert werden, die beliebig komplexe Bedingungen enthalten darf. Sie bezieht sich auf den jeweils gerade betrachteten Rekursionspfad und hat folgende Semantik: Wenn durch die Hinzunahme eines weiteren Komponentenmoleküls zum aktuellen Pfad die Bedingung erfüllt würde, wird die Berechnung des Pfades abgebrochen. In der UNTIL-Klausel kann auf MAX_RECDEPTH Bezug genommen werden, so daß die Rekursionstiefe jedes Pfades beschränkt werden kann.

Beispiel: *Welche Städte sind von Paris aus mit höchstens einem Zwischenstop erreichbar?*

```
SELECT  ALL
FROM    Start (Stadt) - direkte_Flugverbindung-Stadt
                        RECURSIVE Stadt.Flüge-direkte_Flugverbindung
                        UNTIL MAX_RECDEPTH > 1
WHERE   Start.Name='Paris'
```

Ferner sind die Schlüsselwörter CURRENT und PREVIOUS erlaubt, um das aktuell untersuchte Komponentenmolekül bzw. seinen direkten Vorgänger anzusprechen.

Beispiel: *Welche Städte sind von Paris aus ohne Linienwechsel erreichbar?*

```
SELECT  ALL
FROM    Start (Stadt) - direkte_Flugverbindung-Stadt
                        RECURSIVE Stadt.Flüge-direkte_Flugverbindung
                        UNTIL  direkte_Flugverbindung (CURRENT).Linie <>
                               direkte_Flugverbindung (PREVIOUS).Linie
WHERE   Start.Name='Paris'
```

Bisher wurden nur “lineare” Rekursivmoleküle vorgestellt, also solche, bei denen jedes Atom der Komponentenmoleküle in dem Pfad vom Wurzelatom zum “untersten” Atom des Rekursivmoleküls enthalten war. Daß auch andere Rekursivmoleküle definierbar sind, zeigt folgendes Beispiel: *Welche innerfranzösischen Flugverbindungen gibt es? (Wie kann man durch Frankreich reisen, ohne eine Flugverbindung mehrmals zu benutzen?)*

```
SELECT  ALL
FROM    Start (Stadt) - (direkte_Flugverbindung-Stadt-Staat
                        RECURSIVE Stadt.Flüge-direkte_Flugverbindung
                        UNTIL  Staat (CURRENT).Name <> 'Frankreich');
WHERE   Start.Name='Paris'
```


Die Hinzunahme der jeweils ersten Stadt außerhalb Frankreichs zum Rekursivmolekül würde zur Erfüllung der UNTIL-Klausel führen. Daher werden außerfranzösische Städte nicht in das Rekursivmolekül aufgenommen.

6. Die Berechnung der generalisierten transitiven Hülle

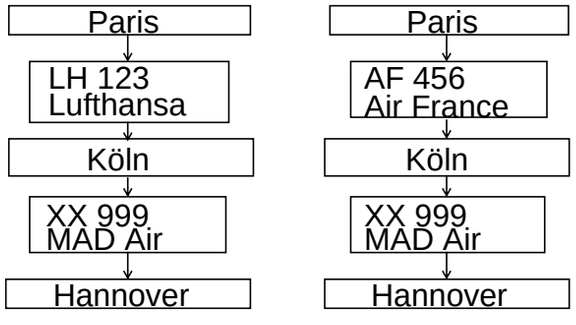
Die bisher eingeführten Rekursivmoleküle erlauben die Berechnung der transitiven Hülle. Zu jedem Atom der transitiven Hülle sind alle (bzgl. der Anzahl der zu traversierenden Beziehungen) minimalen Wege in Form von Strukturinformation im Ergebnis enthalten. Zur Berechnung der generalisierten transitiven Hülle, also zur Auswahl bestimmter Pfade über die Pfadeigenschaften, eignet sich das Ergebnis der Anfrage in dieser Form jedoch nicht.

Daher wird neben der RECURSIVE-Klausel eine weitere Klausel **REC_PATH** angeboten. Tritt diese in einer Anfrage auf, so werden statt der transitiven Hülle alle entsprechenden maximalen schleifenfreien linearen Pfade als Moleküle geliefert. Das heißt, daß nicht alle Werte des rekursionsbildenden REFERENCE-Attributes zum Aufbau des Rekursivmoleküles benutzt werden, sondern jeweils nur einer. Wo RECURSIVE also ein Molekül liefert, das die transitive Hülle (oder einen Ausschnitt daraus) enthält, liefert REC_PATH für jeden Pfad, der über die rekursivmolekülbildende Beziehung spezifiziert wird, ein eigenes Molekül. Die Komponentenmoleküle werden dabei einer speziellen qualifizierten Projektion unterworfen: alle Atome des Komponentenmoleküls gehören nur dann zum Rekursivmolekül, wenn ihr Atomtyp nicht zur Rekursionsbildung beiträgt (wie etwa Staat im vorhergehenden Beispiel), oder wenn die Atome direkt zu dem Rekursivpfad gehören, den das Rekursivmolekül darstellt.

Beispiel: *Welche Stadt- und Linienkombinationen gibt es für den Flug ab Paris (wenn keine Stadt mehrmals erreicht werden soll)?*

```
SELECT  ALL
FROM    Stadt-direkte_Flugverbindung
                REC_PATH Flugverbindung.zwischen-Stadt
WHERE Stadt(0).Name = 'Paris'
```

Der Ausschluß von direkte_Flugverbindung (LAST) ist aufgrund der Semantik der Rekursivmolekülbildung hier nicht nötig. Als Ergebnis entstehen folgende zwei Moleküle:



Mit dieser REC_PATH-Klausel kann nun die sogenannte generalisierte transitive Hülle berechnet werden. Tabelle 2 zeigt die im MAD-Modell zur Berechnung der einzelnen Problemtypen (vgl. Tabelle 1) zu verwendenden Operatoren. *MOLAGG* und *VALUE* liefern - wie in Kapitel 4 bereits erläutert - jeweils Listen von Werten, auf die dann die Operationen MIN, SUM, etc. angewendet werden können. Der Operator *MULT* bildet das Produkt aller Listenelemente.

Problemtyp	Gesuchte Eigenschaft	Aggregations-operator	Verkettungs-operator
Kürzester Pfad	Länge oder Dauer	MIN (VALUE...)	SUM (MOLAGG...)
Kritischer Pfad	Länge oder Dauer	MAX (VALUE...)	SUM (MOLAGG...)
Breitester Pfad	Kapazität	MAX (VALUE...)	MIN (MOLAGG...)
Zuverlässigster Pfad	Verfügbarkeit	MAX (VALUE...)	MULT (MOLAGG...)
Stückliste	Häufigkeiten	SUM (VALUE...)	MULT (MOLAGG...)
Alle Wege	Teilstrecken		MOLAGG
Irgendein Weg	Teilstrecken		MOLAGG
Logische Deduktion	Prädikate		MOLAGG

Tabelle 2: Operatoren des MAD-Modells für die Berechnung von Pfadproblemen

Für “irgendein Weg” ist kein spezieller Aggregationsoperator notwendig, da jedes Molekül einen solchen Weg darstellt. Analog kann für die logische Deduktion argumentiert werden. Die Ergebnismenge besteht aus allen Wegen. Daher kann ein Aggregator für “alle Wege” entfallen.

Ein Beispiel für den Problemtyp “kürzester Pfad” ist die folgende Anfrage: Wie lange sind die Passagiere, die in Paris mit Ziel Hannover starten, mindestens in der Luft? (Dazu muß es natürlich ein Attribut Dauer bei dem Atomtyp direkte_Flugverbindung geben.)

```
                                                    Aggregation
SELECT  Start (In_der_Luft :=MIN (VALUE (SELECT Flugdauer FROM RESULT)))
FROM    (SELECT Start (Name),
                (SELECT  Flugdauer := SUM (MOLAGG                          }
                              (direkte_Flugverbindung (ALL_REC) .Dauer)))  } Ver-
                FROM     RESULT                                            } kettung
                WHERE    Stadt (LAST).Name = ’Hannover’)                   }
         FROM   Start (Stadt) - direkte_Flugverbindung-Stadt
                       REC_PATH Stadt.Flüge-direkte_Flugverbindung
                       UNTIL Stadt (PREVIOUS) = ’Hannover’
         WHERE  Start.Name = ’Paris’)
```

In einer inneren SELECT-Anweisung werden die Werte innerhalb eines Pfades verkettet, falls dieser in Hannover endet, um die Gesamtflugdauer zu berechnen. **UNTIL Stadt (PREVIOUS) = ’Hannover’** bewirkt, daß Hannover noch zu dem Pfad dazugehört. In der äußeren SELECT-Anweisung wird dann das Minimum der Gesamtflugdauer aller Pfade Paris-Hannover bestimmt. Sollen nur sinnvolle Flugverbindungen betrachtet werden (Weiterflug nur mit einer Verbindung, die zeitlich später liegt), so muß die UNTIL-Klausel ergänzt werden, etwa um den Ausdruck: **Abflug (CURRENT) > Ankunft (PREVIOUS).**

An diesem Beispiel wird die Nützlichkeit parametrisierter Anfragen, wie MAD sie unterstützt, deutlich: Statt Hannover bzw. Paris könnte in dieser Anfrage auch ein symbolischer Parameter $ZIEL oder $START stehen. Für diese Parameter können dann bei der Ausführung der vorübersetzten Anfrage aktuelle Werte angegeben werden. Aus Platzgründen kann dieser Aspekt hier leider nicht vertieft werden.

7. Beispiele für komplexere Rekursionsbeziehungen

Mehrfachrekursion

Für die folgenden Betrachtung führen wir eine Erweiterung des Beispielschemas ein:

```
CREATE_ATOM_TYPE Stadt :
    (Stadt_id : IDENTIFIER;
    Name : String;
    Nation : REFERENCE TO Staat.Städte (1,1)
    Flüge : REFERENCE TO direkte_Flugverbindung.zwischen (0,VAR);
    Züge : REFERENCE TO Zug.zwischen (0,VAR);
    ...)

CREATE_ATOM_TYPE Zug:
    Zug_Id : IDENTIFIER;
    Zug_Nr : INTEGER;
    zwischen: REFERENCE TO Stadt.Züge (2,2)
    ...)
```

Die Frage: *Wie gelangt man von Kaiserslautern mit dem Zug zu einem Flughafen und per Flugzeug nach Paris?* bedingt eine Verbindung der Rekursivmoleküle für Flugverbindung und Zugverbindung. Dies kann durch Verbindung (Join) zweier Rekursivmoleküle erreicht werden, der die Zugverbindung von einer Stadt X nach Kaiserslautern mit der Flugverbindung von X nach Paris verbindet (Voraussetzung: Zug- und Flugverbindung sind symmetrisch):

```
SELECT  ALL
FROM    Bahn (Stadt-Zug REC_PATH Zug.zwischen-Stadt
                    UNTIL Stadt (PREVIOUS).Name='Kaiserslautern'
        Flug (Stadt - direkte_Flugverbindung
                    REC_PATH direkte_Flugverbindung.zwischen-Stadt
                    UNTIL Stadt (PREVIOUS).Name='Paris')
WHERE   Bahn.Stadt(0).Name=Flug.Stadt (0).Name
        AND Flug.Stadt (LAST).Name = 'Paris'
        AND Bahn.Stadt (LAST).Name='Kaiserslautern'
```


Ein weitere Variante "verschachtelt" die Rekursivmoleküle ineinander. Für jede per Zug erreichbare Stadt wird untersucht, ob sie eine Flugverbindung nach Paris hat. Die unterstrichene UNTIL-Klausel bewirkt, daß die Bahnverbindung von dieser Stadt aus nicht weiter verfolgt wird, wenn eine entsprechende Flugverbindung exisitert.

```
SELECT  ALL
FROM    Start(Stadt) - Zug - Stadt -
                    Flug :=  (SELECTALL
                             FROM   Flug(direkte_Flugverbindung - Ziel(Stadt)
                                    REC_PATH Ziel.Flüge - direkte_Flugverbindung
                                    UNTIL Ziel (PREVIOUS).Name='Paris')
                             WHERE  Ziel(LAST).Name = 'Paris')
                    REC_PATH Stadt.Züge - Zug
                    UNTIL EXISTS Flug (PREVIOUS)
WHERE Start.Name = 'Kaiserslautern'
```

7

Rekursion über mehrere REFERENCE-Attribute

Ein vielverwendetes Beispiel zur Demonstration von Rekursion, insbesondere im Bereich deduktiver Datenbanken, ist das Vorfahren-Problem: Zu einer gegebenen Person sind alle Vorfahren gesucht.

Wird das MAD-Modell-Schema folgendermaßen gewählt:

```
CREATE ATOM_TYPE Person:
    (Id : IDENTIFIER;
    Name : String;
    Mutter : REFERENCE TO Person.ist_Mutter_von (0,1);
    Vater : REFERENCE TO Person.ist_Vater_von (0,1);
    ist_Mutter_von: REFERENCE TO Person.Mutter (0,VAR);
    ist_Vater_von: REFERENCE TO Person.Vater (0,VAR);
    ...)
```

so müssen zur Bestimmung aller Vorfahren die Referenzen auf Mutter und Vater vereinigt werden (& ist der Konkatenationsoperator für REFERENCE- und LIST-Attribute.):

```
SELECT  ALL
FROM    P:=( SELECT Name, Vorfahren:=Mutter&Vater
             FROM Person)
        RECURSIVE P.Vorfahren-P
WHERE   P(0).Name = 'Hugo'
```

8

8. Zusammenfassung und Ausblick

Die Integration von Rekursion in ein Datenmodell ist ein wichtiger Schritt auf dem Weg zu Datenbanksystemen für Non-Standard-Anwendungen. Erwünscht ist dabei nicht nur die Fähigkeit zur Berechnung der transitiven Hülle, sondern auch einer generalisierten transitiven Hülle, um die wichtige Rekursionsklasse der Pfadprobleme behandeln zu können. Im Molekül-Atom-Datenmod-

ell können die transitive Hülle und die generalisierte transitive Hülle berechnet werden, so daß Pfadprobleme auf azyklischen Graphen gelöst werden können. Mechanismen anderer Datenmodelle zur Integration einer solchen Fähigkeit sehen meist nur eine flache Sicht der transitiven Hülle vor, also nur die Erfassung aller Objekte innerhalb dieser Hülle. Das MAD-Modell bietet darüberhinaus die Möglichkeit, diese transitive Hülle - ebenso wie einzelne Pfade - in ihrer Struktur darzustellen, und zwar als Objekt innerhalb des Datenmodells. Damit ist es auch möglich, Pfade darzustellen, deren einzelne Knoten wiederum strukturiert sind. Rekursion läßt sich im MAD-Modell deskriptiv spezifizieren, so daß der Anwender von der Erstellung eines effizienten Abarbeitungsplanes befreit wird.

Die Implementierung des MAD-Modells im Projekt PRIMA (Prototyp-Implementierung des MAD-Modells, /HMMS87, Hä88/) enthält die Realisierung der RECURSIVE-Klausel. Eine in Vorbereitung befindliche zweite Version von PRIMA wird den vollen Umfang des hier vorgestellten Sprachvorschlags realisieren.

Literatur

BTW85 Proceedings Datenbank-Systeme für Büro, Technik und Wissenschaft, GI-Fachtagung, Karlsruhe, März 1985, Springer IFB 94.

BTW87 Proceedings Datenbank-Systeme für Büro, Technik und Wissenschaft, GI-Fachtagung, Darmstadt, April 1987, Springer IFB 136.

Da86 Dadam, P. et al.: A DBMS Prototype to Support Extended NF^2-Relations: An Integrated View on Flat Tables and Hierarchies, in: Proc. ACM SIGMOD, Washington, DC, 1986, pp. 356-367.

De87 Demo, B.: Recursive versus Iterative Schemes for Least Fix Point Computation in Logic Databases, in: Proc. Third Int. Conf. on Data Engineering, Los Angeles, California, Feb. 3-5, 1987, pp. 130-137.

DS86 Dayal, U., Smith, J.M.: PROBE: A Knowledge-Oriented Database Management System, in : Brodie, M.L., Mylopoulos, J. (eds.): On Knowledge Base Management Systems, Springer, 1986, pp 227-257.

Hä85 Härder, T., et al.: Datenstrukturen und Datenmodelle für den VLSI-Entwurf, Forschungsbericht 26/85 des SFB 124, Universität Kaiserslautern, 1985.

Hä88 Härder, T. (ed.): The PRIMA Project - Design and Implementation of a Non- Standard Database System, Forschungsbericht 26/88 des SFB 124, Universität Kaiserslautern, 1988.

HMMS87 Härder, T., Meyer-Wegener, K., Mitschang, B., Sikeler, A.: PRIMA - A DBMS Prototype Supporting Engineering Applications, in: Proc. 13. Int. Conf on VLDB, Brighton, 1987, pp. 433-442.

HN84 Henschen, L.J., Naqvi, S.A.: On Compiling Queries in Recursive First-Order Databases, in: Journal of the ACM, Vol. 31, No. 1, January 1984, pp. 47-85.

HR85 Härder, T., Reuter, A.: Architektur von Datenbanksystemen für Non-Standard-Anwendungen, in /BTW85/ , S. 253-286.

Io86 Ioannidis, Y.E.: On the Computation of the Transitive Closure of Relational Operators, in: Proc. 12. Int. Conf on VLDB, Kyoto, August 1986, pp. 403-411.

KCB87 Kim, W., Chou, H-T., Banerjee, J.: Operations and Implementation of Complex Objects, in: Proc. Third Int. Conf. on Data Engineering, Los Angeles, California, Feb. 3-5, 1987, pp. 626-633.

Li87 Linnemann, V.: Non First Normal Form Relations and Recursive Queries: An SQL Based Approach, in: Proc. 3rd IEEE Int. Conf. on Data Engineering, Los Angeles, Feb. 1987, pp. 591-598.

LMR87 Lu, H., Mikkilineni, K., Richardson, J.P.: Design and Evaluation of Algorithms to Compute the Transitive Closure of a Database Relation, in: Proc. Third Int. Conf. on Data Engineering, Los Angeles, California, Feb. 3-5, 1987, pp. 112-119.

Mi85 Mitschang, B.: Charakteristiken des Komplex-Objekt-Begriffs und Ansätze zu dessen Realisierung, in: /BTW85/, S. 382-400.

Mi88 Mitschang, B.: Ein Molekül-Atom-Datenmodell für Non-Standard-Anwendungen -Anwendungsanalyse, Datenmodellentwurf und Implementierungskonzepte, Dissertation, Universität Kaiserslautern, 1988.

Re87 Reuter, A.: Kopplung von Datenbank- und Expertensystemen, in: Informationstechnik it, 29. Jg., Heft 3/87, S. 164-175.

RS86 Raschid, L., Su, S.Y.W.: A Parallel Processing Strategy for Evaluating Recursive Queries, in: Proc. 12. Int. Conf on VLDB, Kyoto, August 1986, pp. 412-419.

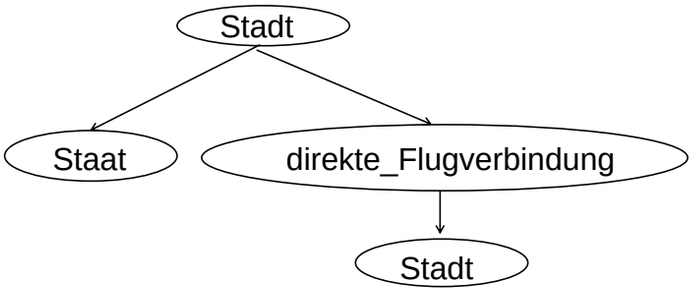
SR86 Stonebraker, M., Rowe, L.A.: The Design of POSTGRES, in: Proc. ACM SIGMOD, Washington, DC, 1986, pp. 340-355.

SS86 Scheck, H.J., Scholl, M.H.: The Relational Model with Relation-Valued Attributes, in: Information Systems, Vol. 11, No. 2, 1986, pp. 137-147.

Ul85 Ullman: Implementation of Logical Query Languages for Databases, in: ACM TODS, Vol. 10, No. 3, pp. 289-321.

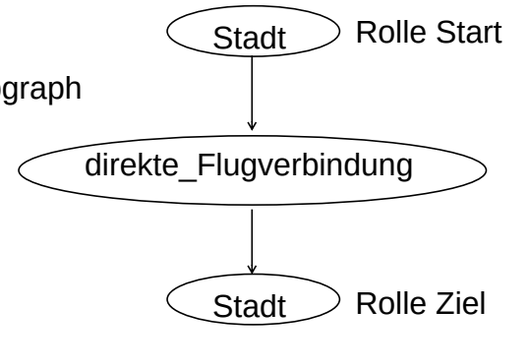
Anhang

1 Molekültypgraph



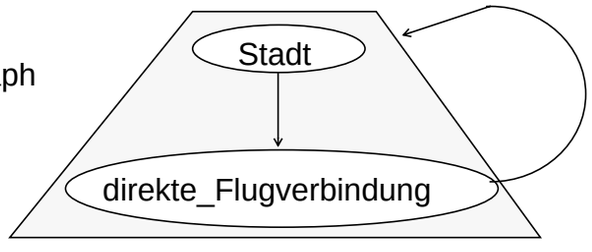
---

2 Molekültypgraph

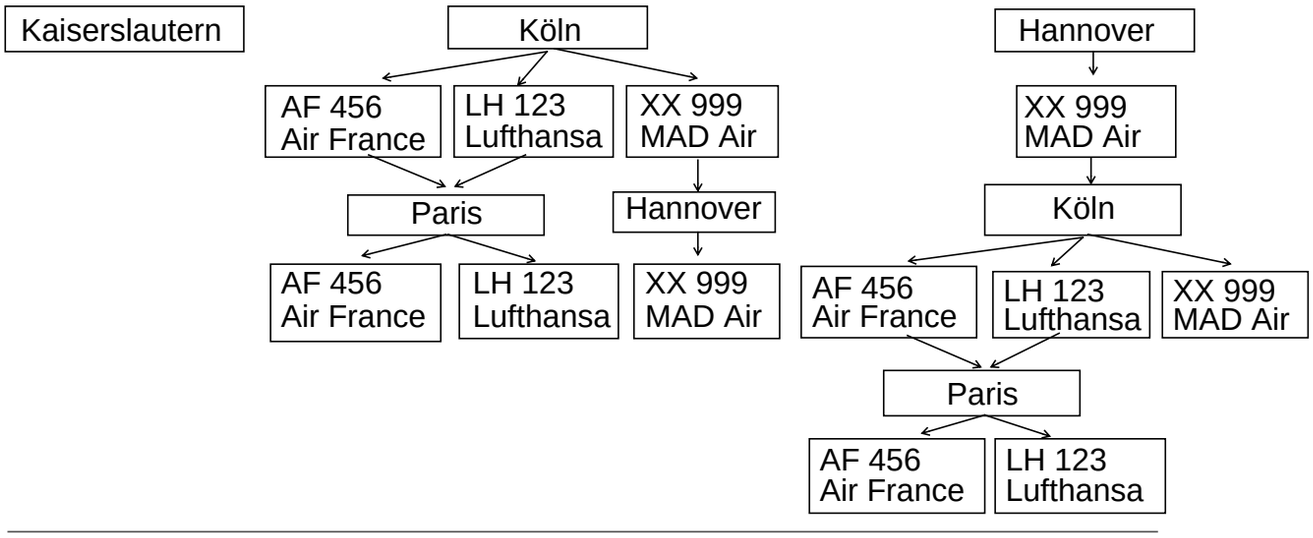


---

3 Molekültypgraph

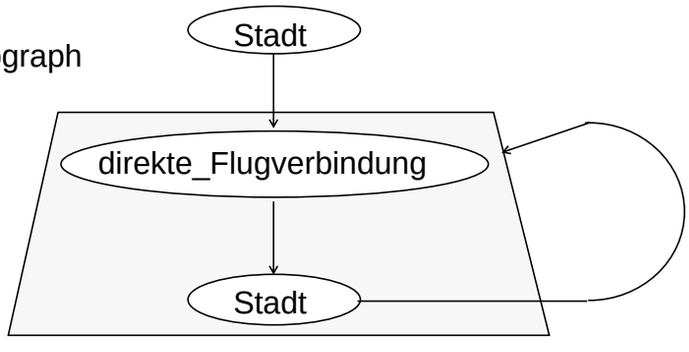


neben dem vorne gezeigten Molekül gehören folgenden drei Moleküle zum Ergebnis:



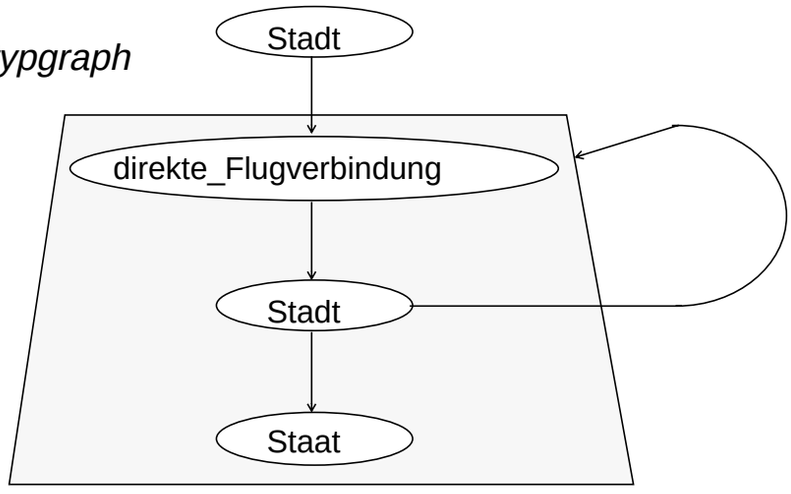
---

4 Molekültypgraph



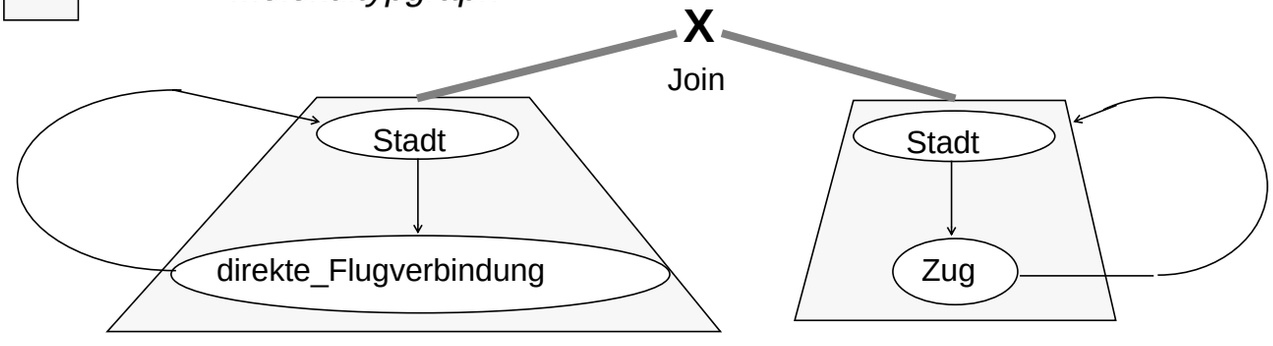
5

Molekültypgraph



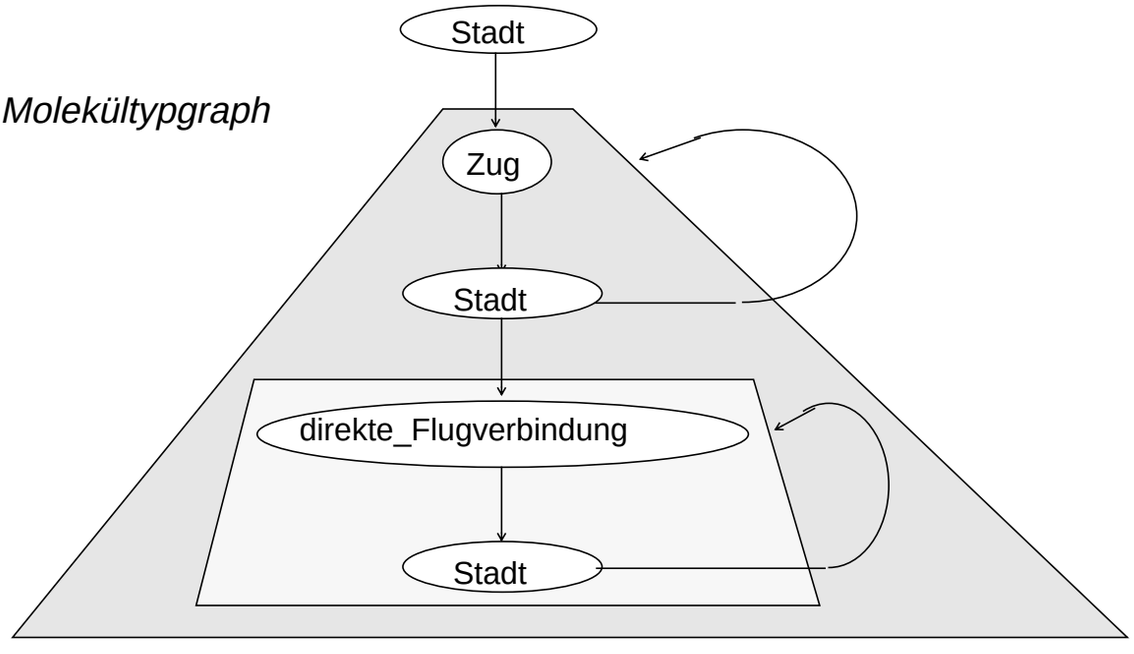
6

Molekültypgraph



7

Molekültypgraph



8

Molekültypgraph