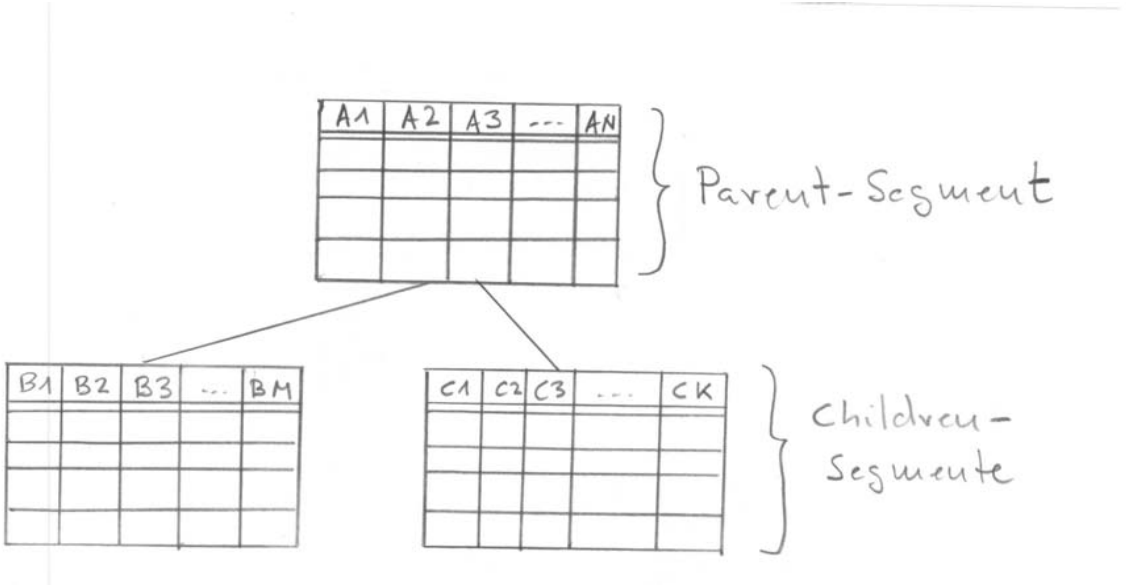
4. Hierarchische und netzwerkartige Datenbankmodelle

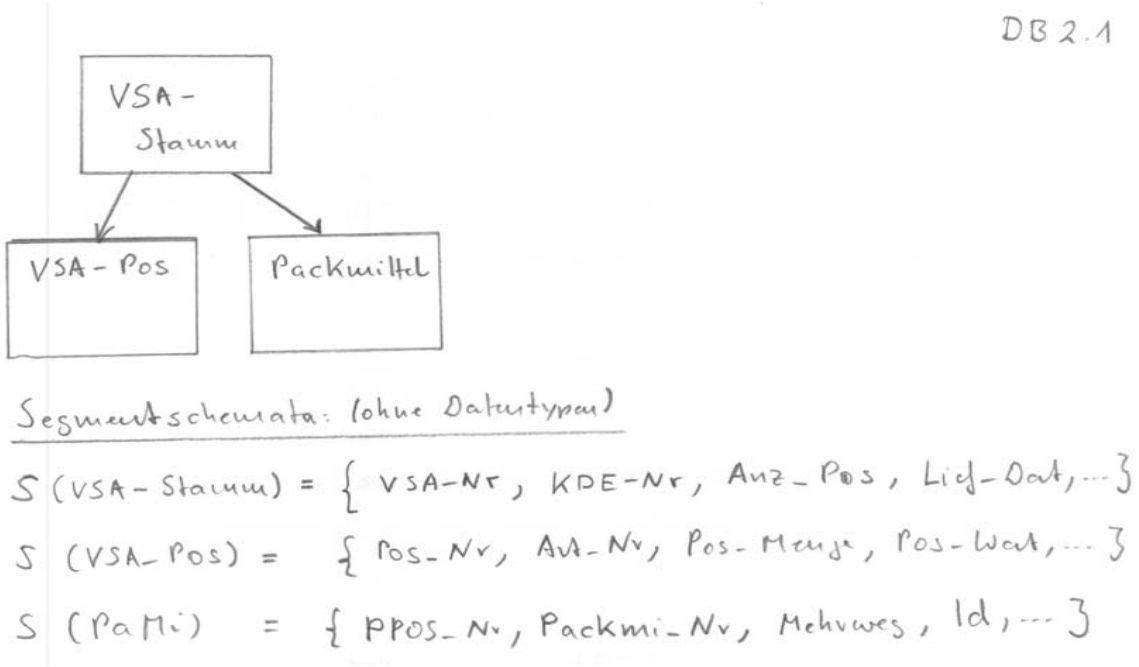
4.1 Hierarchische Datenbanken

Hierarchien können durch Baumgraphen beschrieben werden. Datensätze einer hierarchischen Datenbank (HDB) sind in Segmenten organisiert. Segmente enthalten Datensätze gleichen Aufbaus.

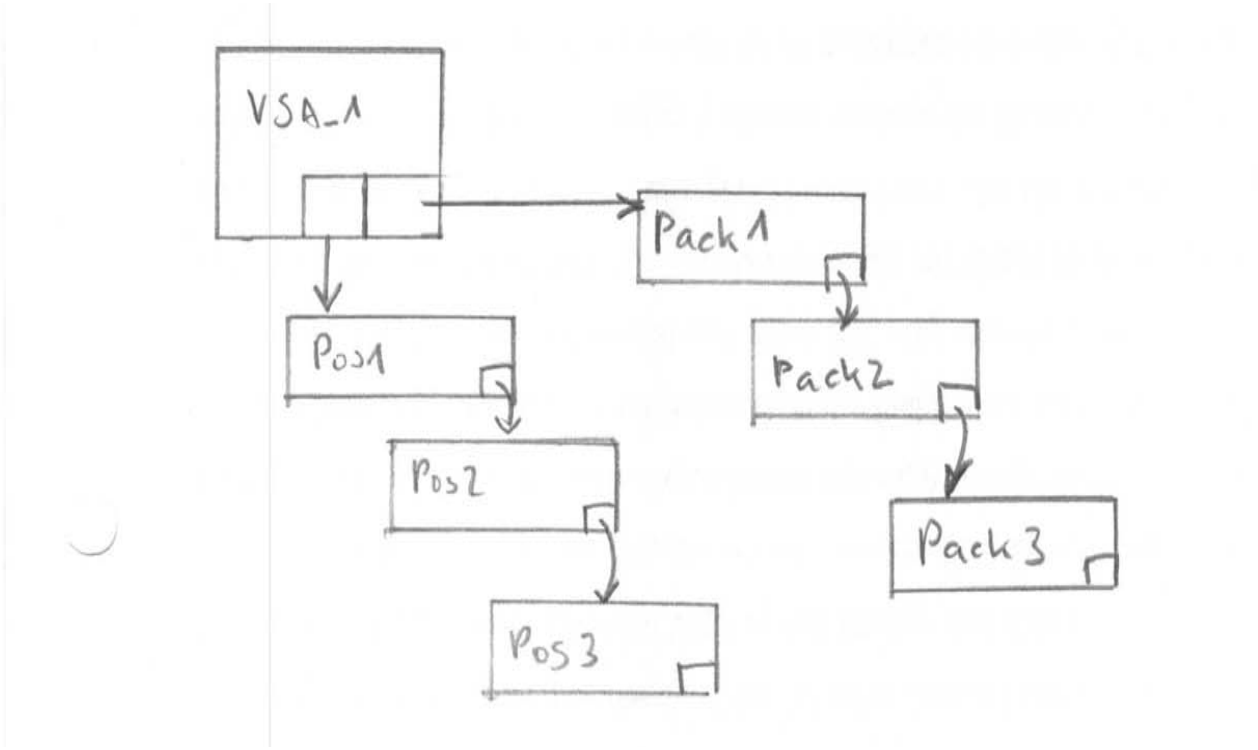


Logische Abhängigkeiten in Form von Hierarchien werden in einer HDB dadurch verwaltet, dass Children-Segmente einem Parent-Segment untergeordnet werden. Die Segmente stehen in einer baumartigen Anordnung.

BSP.1: Eine Versandauftragsdatenbank (VSA) mit den Segmenten **VSA-Stamm** (Parent) und den Children-Segmenten **VSA-Position** (VSA-Pos) und **Packmittel**.



Die Hierarchie wird durch Pointer-Ketten (Listen) implementiert, in denen jeweils ein Parent-Datensatz auf eine Liste von Children-Datensätze zeigt.



Eine Anfragesprache für hierarchische Datenbanken vom Typ IMS® ist die Sprache DLI (Data Language Interface). Diese baut auf der hierarchischen Struktur auf:

Einfügeoperation: **ISRT** : notwendige Parameter: (DS, E_Seg [, P_Seg, PRIK_Parent]) mit:

- DS = einzufügender Datensatz,
- E_Seg = Name des Segments, in das der DS eingefügt werden soll,
- P_Seg = Name des Parentsegmentes,
- PRIK_Parent = PRIK-Wert des übergeordneten Parent-Datensatzes

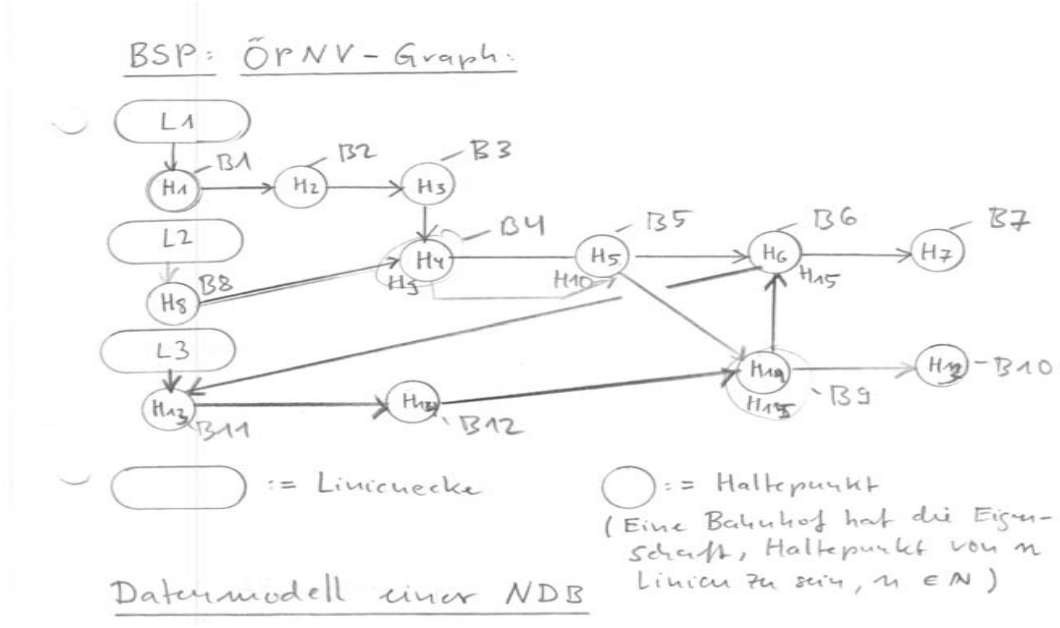
Leseoperationen: a) Direktzugriff auf einen Datensatz in einem Segment (**GetUnique**) Parameter: Segmentname, PRIK-Segment
b) Sequentielles Weiterlesen im Segment nach einem erfolgreichen GetUnique (**GetNext**) Parameter: Segmentname
c) Lesen aller Datensätze in einem Children-Segment, die einem Parentdatensatz untergeordnert sind (**GetNext within Parent**) Parameter: PRIK_Parent, Ch_Segmentname

4.2. Netzwerkartige Datenbanken (vgl. Vossen, S.83-114)

Netzwerkartige Datenbanken sind standardisiert worden durch CODASYL (Conference on Data Systems Language). Das NW-Datenbankmodell besteht aus folgenden Elementen: (1) **Segmente**: Datensätze gleicher Art werden jeweils in einem Segment

organisiert. (Bsp.: Segment = LINIE, Segment = BHF (Bahnhof), Segment = Hpkt (Haltepunkt)

(2) Sets (Mathematisch: Mengen): 1 Datensatz eines Segmentes A wird mit **n** Datensätzen eines Segmentes B verknüpft sein. (Bsp.: Set1: „Linie verbindet“: 1 Linie verbindet n Bahnhöfe. Set2: „ist Haltepunkt von“: 1 Bahnhof ist Haltepunkt von n Linien.



Ein modernes DBMS, das **netzwerkartige Graphen** verwalten kann und somit ein NWDBMS (netzwerkartiges DBMS) ist, ist **Neo4J**. Neo4J ist im Zusammenhang mit der neueren Diskussion um NoSQL-Datenbanken bekannt geworden. NoSQL steht für „not only SQL“. Bei NoSQL-Datenbanken werden Datenmodelle betrachtet, die entweder nicht relational oder über das relationale Modell hinausgehend sind. Neo4J hat als Datenmodell das Modell eines persistenten **Graphen**.

Ein Graph **G** ist ein Verbund einer Menge **P** von Punkten (Ecken) und einer Menge **K** von Kanten: **G = (P, K)**. Hierbei verbindet jede Kante **k** zwei Ecken **p1** und **p2**.

BSP.1: In einer Aufgabe des nächsten Praktikumsversuchs soll aus mehreren RDB Tabellen Informationen zum Aufbau eines Graphen **Gr** mit Hilfe eines JDBC-Programms **JP1** zusammengestellt werden. **JP1** greift hierzu lesend auf die RDB Tabellen zu und erzeugt Informationen über die Ecken p und die Kanten k des in der Neo4J zu speichernden Graphen **Gr**. Diese Informationen werden von JP1 zeichenorientiert in eine zu erzeugende XML-Datei **Gr.XML** geschrieben (s. Abb.1: DFP für JP1). Das Neo4J-DBMS des Informatik-Labors verfügt über einen Import-Modul, der prüft, ob die Datei **Gr.XML** in Hinsicht auf den zu speichernden Graphen **valide** ist. Ist die Datei valide, wird der in **Gr.XML** beschriebene Graph gespeichert, sonst bekommt der Anwender eine Fehlermeldung.

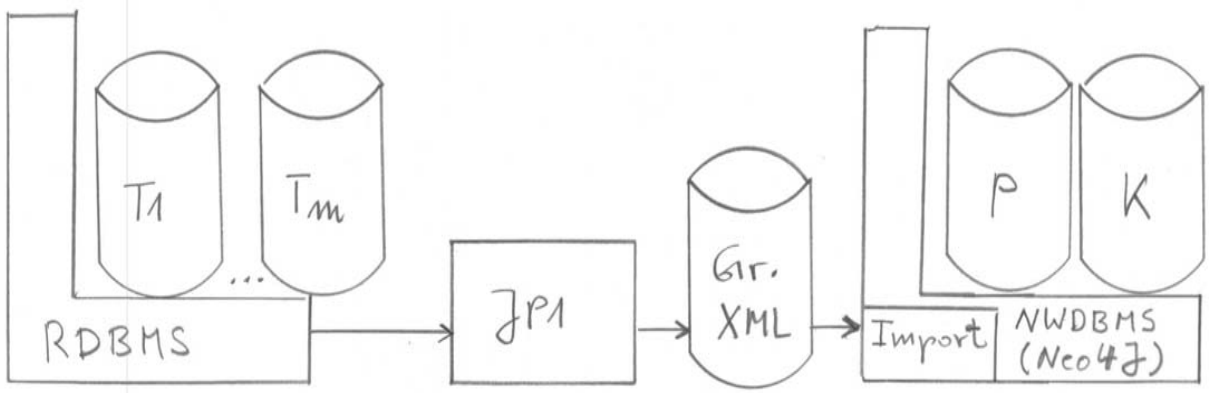


Abb.1: DFP für JP1

BSP.2: Eine XML-Datei, die einen Graphen G = (P,K) enthält, wird aus einer Tabelle RORT, die Ortsinformationen enthält (1 Ort = 1 Ecke), und einer Tabelle ROUTPLAN, die Streckeninformationen enthält (1 Strecke = 1 Kante), erzeugt.

a) Inhalt von RORT:

```
"PLZ"    "ORT"
53111    Bonn
50678    Koeln
```

```
53842    Troisdorf
52064    Aachen
40212    Duesseldorf
```

b) Inhalt von ROUTPLAN:

"ROUT"	"STRECK"	"APLZ"	"BPLZ"	"KM"
BNDUE1	1	53111	50678	30
BNDUE1	2	50678	40212	40
BNDUE1	3	40212	53111	70
BTDFAC1	1	53111	53842	10
BTDFAC1	2	53842	50678	20
BTDFAC1	3	50678	52064	60
BTDFAC1	4	52064	53111	90

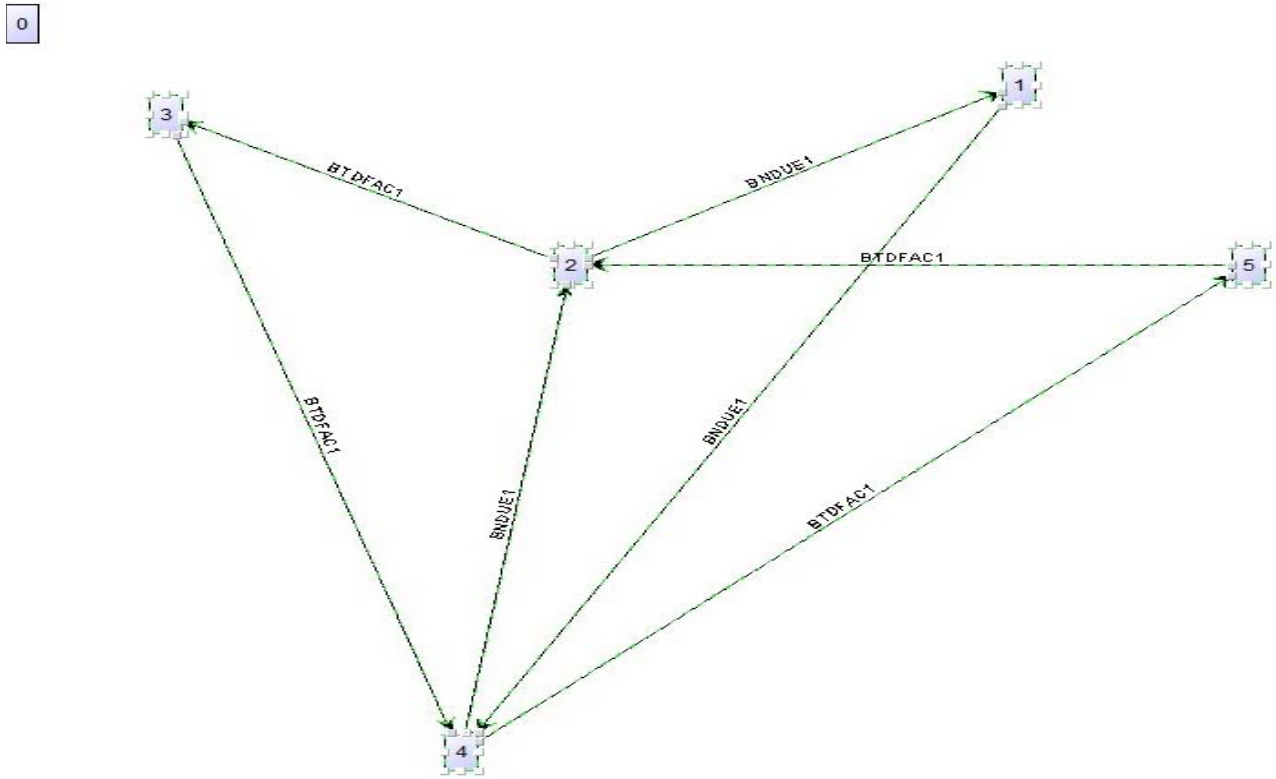
Die mit einem JDBC-Programm erzeugte XML-Datei (hier ROUT60.XML) hat folgenden Inhalt:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<graph xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<!--Datenbankname-->
<dbname>RoutenplanMHENKDB</dbname>
<!--Knoten mit Attributen-->
<node nodeId="4">
<attribute attName="ORT" attType="neo:string">Bonn</attribute>
```

```
<attribute attName="PLZ" attType="neo:int">53111</attribute>
</node>
<node nodeId="1">
<attribute attName="ORT" attType="neo:string">Duesseldorf</attribute>
<attribute attName="PLZ" attType="neo:int">40212</attribute>
</node>
<node nodeId="3">
<attribute attName="ORT" attType="neo:string">Aachen</attribute>
<attribute attName="PLZ" attType="neo:int">52064</attribute>
</node>
<node nodeId="5">
<attribute attName="ORT" attType="neo:string">Troisdorf</attribute>
<attribute attName="PLZ" attType="neo:int">53842</attribute>
</node>
<node nodeId="2">
<attribute attName="ORT" attType="neo:string">Koeln</attribute>
<attribute attName="PLZ" attType="neo:int">50678</attribute>
</node>
<edge edgeId="1" type="directed" source="4" target="2" weighting="30"
label="BNDUE1    ">
<attribute attName="STRECK" attType="neo:int">1</attribute>
</edge>
<edge edgeId="2" type="directed" source="2" target="1" weighting="40"
label="BNDUE1    ">
<attribute attName="STRECK" attType="neo:int">2</attribute>
</edge>
<edge edgeId="3" type="directed" source="1" target="4" weighting="70"
label="BNDUE1    ">
<attribute attName="STRECK" attType="neo:int">3</attribute>
</edge>
```

```
<edge edgeId="4" type="directed" source="4" target="5" weighting="10"
label="BTDFAC1   ">
<attribute attName="STRECK" attType="neo:int">1</attribute>
</edge>
<edge edgeId="5" type="directed" source="5" target="2" weighting="20"
label="BTDFAC1   ">
<attribute attName="STRECK" attType="neo:int">2</attribute>
</edge>
<edge edgeId="6" type="directed" source="2" target="3" weighting="60"
label="BTDFAC1   ">
<attribute attName="STRECK" attType="neo:int">3</attribute>
</edge>
<edge edgeId="7" type="directed" source="3" target="4" weighting="90"
label="BTDFAC1   ">
<attribute attName="STRECK" attType="neo:int">4</attribute>
</edge>
</graph>
```

Aus dieser XML-Datei kann das IMPORT-Werkzeug der Graphdatenbank Neo4J einen Graphen G = (P,K) erzeugen, der in dem VIEW-Werkzeug auch direkt anschaulich als Graph dargestellt werden kann:



(Abb.2: Graph G = (P,K) eines Routenplans)