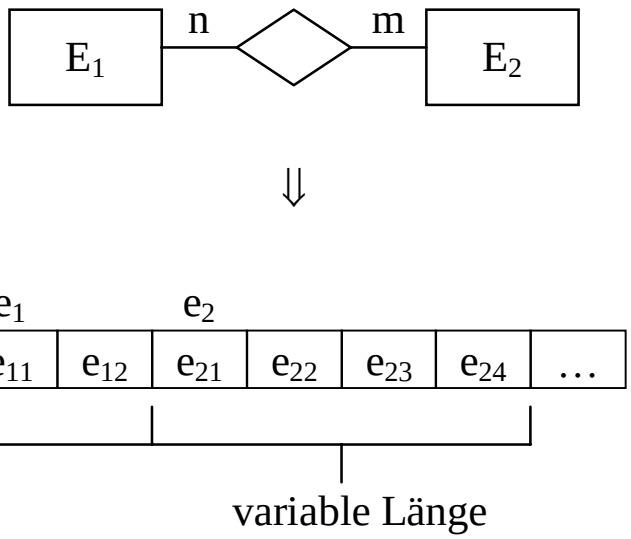
SÄTZE MIT VARIABLER LÄNGE

Generellere Struktur von Datensätzen:

- Ersetzen von Felder durch Gruppen von Feldern.

Beispiel:



→ **Logische Datei** mit Sätzen variabler Länge

Definition von Sätzen mit variabler Länge

1. Ein Satz mit variabler Länge ist eine Liste von „Elementen“.

2. Ein Element ist entweder ein einfaches Feld oder eine variable Gruppe mit Null oder mehr Sätzen variabler Länge.

Notation:

Wie bei regulären Ausdrücken,

$(\alpha)*$ ist eine Gruppe mit mehreren Elementen

$\alpha_1\ \alpha_2\ \alpha_3 \ldots \alpha_n$

Beispiel:

Datei mit (amerikanischen Bundes-) Staaten und der Interstate-Highways innerhalb der Staaten.

Struktur:

STATE (HIGHWAY)*

r_1	New Jersey	178	180	195	
r_2	Alaska				
r_3	North California	148	180	160	190

Erweitertes Beispiel:

STATE POPULATION (HIGHWAY)*

r_1	New Jersey	7.000.000	178	180	195
r_2	Alaska	800.000			

Beispiel: (Fortsetzung)

STATE POPULATION (HIGHWAY
LENGTH (TERMINAL)*)*

r_1	Hessen		6.000.000		
	A66	100	Wiesbaden	Frankfurt	…
	A3	120	Kassel	Darmstadt	…
r_2	Bayern		…		

Wie kann dies in einem Block gespeichert werden?

1. Die Information über den Datentyp jedes Elements muß vorhanden sein.
   Diese Information kann entweder im Blockheader abgelegt oder mit jedem Feld mitgeführt werden.

2. Jeder Satz mit variabler Länge kann durch einen oder mehrere Sätze fester Länge repräsentiert werden, dafür gibt es drei verschiedene Ansätze:

   a) Reserved Space Method.

   b) Pointer Method.

   c) Combined Method (a + b).

RESERVED SPACE METHOD

Annahme: Es gibt eine obere Grenze c von auftretenden Wiederholungen in einer Gruppe.

$$\text{WERT\_A} \left(\overset{\max=c}{\text{WERT\_}B}\right)^*$$

Dann kann die sich wiederholende Gruppe durch c Gruppen von Feldern ersetzt werden.

$$\text{WERT\_A} \quad \underbrace{\text{WERT\_}B \ \ldots \ \text{WERT\_}B}_{c}$$

Falls weniger als c Wiederholungen auftreten, dann

- Werden die fehlenden Wiederholungen durch **Null**-Werte ersetzt, oder
- ein **Zähler**, der die tatsächlich auftretenden Wiederholungen zählt, wird geführt.

Beispiel:

STATE POPULATION

(HIGHWAY LENGTH $(\text{TERMINAL})^{\max = 2}$ *)*

⇓

STATE POPULATION

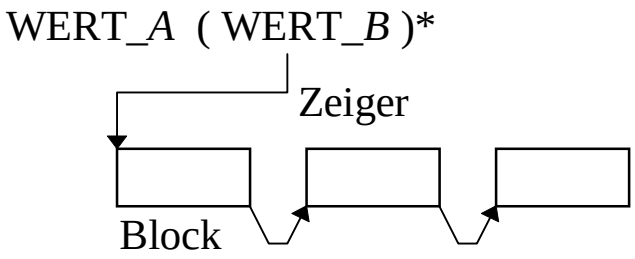
(HIGHWAY LENGTH TERMINAL1 $\text{TERMINAL2}^{\max = 10}$)*

⇓

STATE POPULATION COUNT
HIGHWAY1 LENGTH1 TERMINAL1.1
TERMINAL1.2
⋮
HIGHWAY10 LENGTH10 TERMINAL10.1
TERMINAL10.2

POINTER METHOD

Wenn eine obere Grenze für die Anzahl der Wiederholungen nicht angegeben werden kann, dann kann eine Gruppe durch einen Zeiger auf den ersten Block einer Kette von Blöcken mit den Sätzen der Gruppe ersetzt werden:



⇓

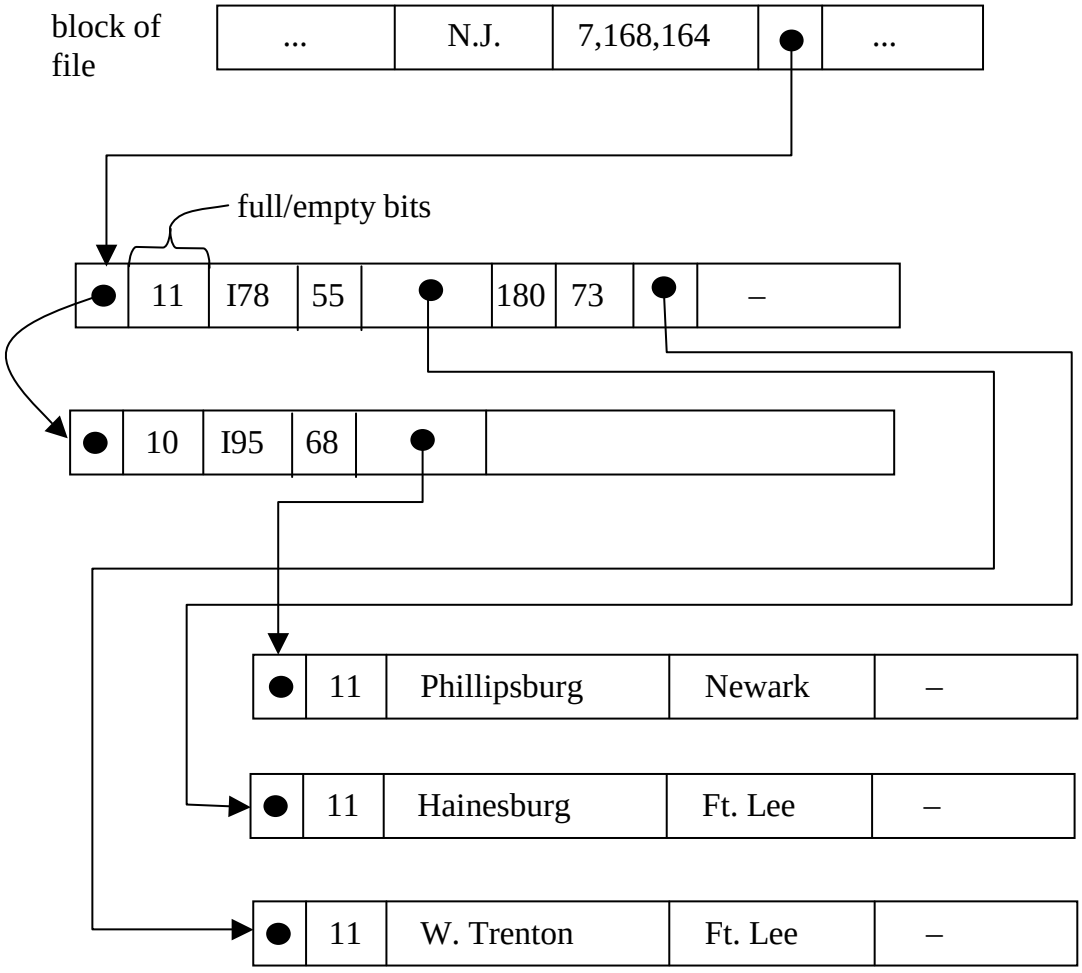
WERT_*A* , P

Beispiel:

Wir nehmen an, unser Highway-Beispiel wird mittels der Pointer-Methode so dargestellt:

STATE POPULATION HPTR
HIGHWAY LENGTH TPTR
TERMINAL

Gespeichert in Blocks sieht das dann so aus:



COMBINED METHOD

- Es können auch die beiden vorangegangenen Methoden miteinander kombiniert werden.

- Es kann z.B. eine Gruppe durch einen Zeiger und eine andere Gruppe durch eine feste Anzahl von Wiederholungen abgebildet werden.

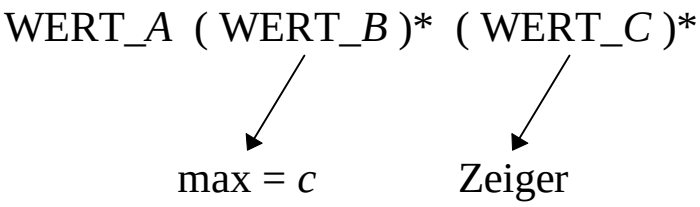
- Eine feste Anzahl von Wiederholungen ist dann vorzuziehen, wenn es eine obere Grenze von Wiederholungen gibt und die durchschnittliche Anzahl von Vorkommen nahe am Maximum liegt.

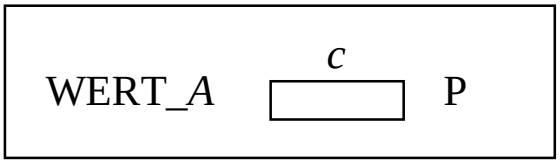
- Wenn das durchschnittliche und das maximale Vorkommen zu sehr voneinander abweichen, ist die Pointer Methode wegen Speicherersparnis vorzuziehen.

- Die Pointer Methode spart Platz, verbraucht aber mehr Blockzugriffe um ein bestimmtes Feld zu finden.

1. Variante der Combined Method:

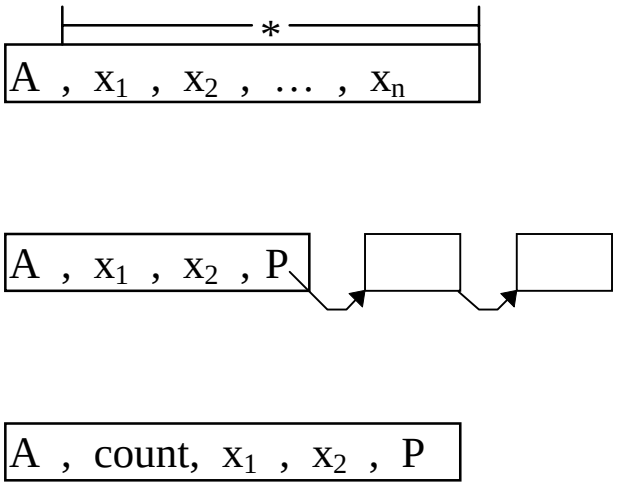
Eine Gruppe mit fester Anzahl, eine andere Gruppe durch die Pointer Methode:





2. Alternative Variante:

Ein Teil Gruppe wird durch eine kleine Anzahl von festen Wiederholungen repräsentiert, dazu kommt ein Zeiger, der auf eventuelle zusätzliche Vorkommen verweist.



Beispiel:

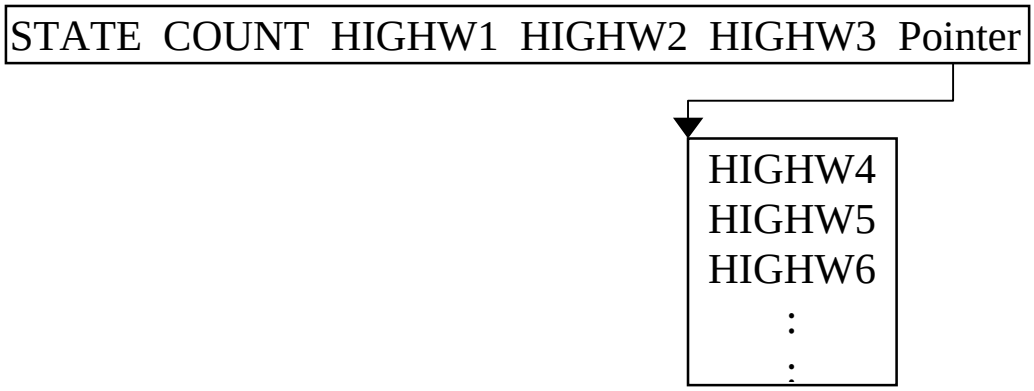
Als Beispiel dienen wieder die Staaten mit ihren Highways.

Annahme:

Es wird Platz für einen Staat und 3 Highways in einem Satz reserviert, weitere Highways sollen per Zeiger adressiert werden.

STATE (HIGHWAY)*

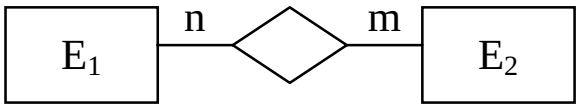
⇓

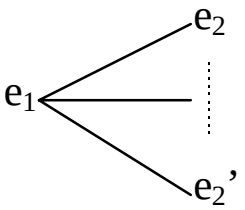


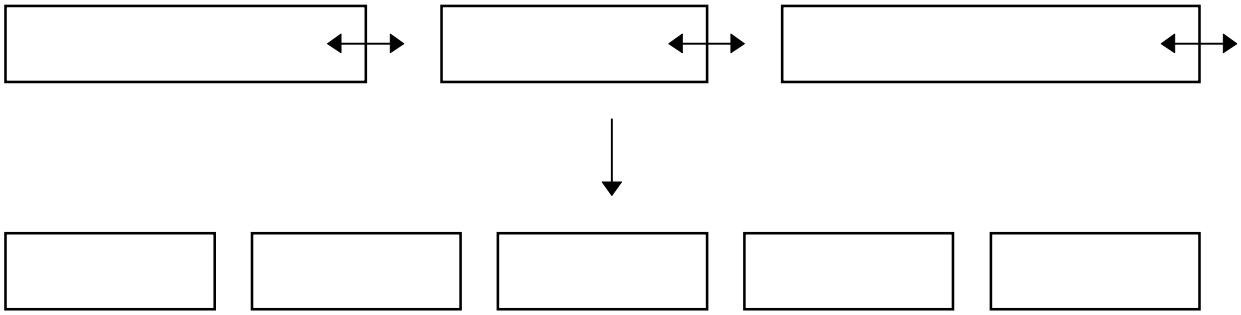
Operationen auf Sätzen mit variabler Länge

Gegeben ist eine logische Datei mit Sätzen variabler Länge, die durch eine Hauptdatei mit Sätzen fester Länge repräsentiert wird (evtl. mit Zeigern auf Blocks, die nicht Teil der Hauptdatei sind).

Einige Felder in dem Teil mit fester Länge formen den Schlüssel für die Sätze der logischen Datei.
Zur Erinnerung: Die Motivation für Sätze variabler Länge war es, many-to-many- oder many-to-one-Beziehungen abbilden zu können.







Die Operationen

- Suchen (Lookup)
- Einfügen (Insertion)
- Löschen (Deletion)
- Ändern (Modification)

funktionieren hier genauso wie bei Sätzen mit fester Länge.

Neue Operationen sind aber die Einfüge, und Löschoperation auf Vorkommen von Feldern **innerhalb einer Gruppe**.

Man kann jede Gruppe als eine eigene **kleine Datei** ansehen. Auf diese Weise können auch hier wieder die bekannten Verfahren angewendet werden.

Im Falle von pinned-down Sätzen innerhalb der Gruppe gelten für das Einfügen und Löschen besondere Regeln, die bereits bei den Hash-Dateien besprochen wurden (Bucket mit pinned Records).

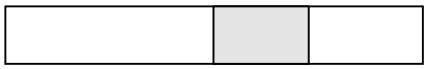
DATENSTRUKTUREN FÜR SUCHE IN FELDERN AUSSERHALB DES SCHLÜSSELS.

Eine Datenbank erlaubt meist auch Anfragen (Queries) in denen in Feldern gesucht wird, die nicht dem Schlüssel angehören.

z.B.:

1. Suche alle Dinosaurier aus dem Jura.
2. Suche nach den Habitaten der fleischfressenden Saurier.
3. Finde alle Dinosaurier die an Land gelebt haben und mindestens 50 Fuß lang waren.

Sekundärindex



r (Record, Satz)

F (Field, Feld)

Domäne von F = D

Ein **Sekundärindex** für das Feld F ist eine Beziehung zwischen der Domäne D und der Menge von Sätzen der Datei.

Ein Sekundärindex stellt sich als eine logische Datei mit Sätzen variabler Länge dar und wird auch als **invertierte Datei** bezeichnet:

WERT (SATZ)*

Eine Instanz von WERT ist ein Wert aus der Domäne D. Eine Instanz von SATZ ist entweder

1. Ein Zeiger auf einen Satz mit dem entsprechenden Wert in Feld F, oder
2. Ein Schlüsselwert für einen Satz mit dem entsprechen Wert in Feld F.

1: (55, p1, p2, p3) 2: (55, Schmidt, Otto, Müller)

1. Falls die erste Option gewählt wird, kann der Zeiger auf

    - einen Block
    - einen Subblock
    - ein Bucket

zeigen. Die Sätze der Hauptdatei sind dann **pinned-down** (!!).

2. Falls die zweite Option gewählt wird, dann sind die Sätze nicht pinned-down, aber es werden weitere Blockzugriffe benötigt, um den entsprechenden Satz zu finden.

SPEICHERSTRUKTUREN FÜR RELATIONEN

Darstellung einer Relation:

Datei, deren Sätze den Tupeln der Relation entsprechen.

HEAP:

Einfachste Struktur: Datei ungeordnet.
Vorteilhaft nur bei sehr kleinen Relationen.

INDEX:

Der Index muß sich nicht unbedingt auf Schlüsselattribute beziehen (Sekundärindex). Damit wird eine Verwendung mehrerer Indizes auf eine Datei möglich
(Achtung!! → pinned Records!).

Mögliche Indexarten:

- Index mit pinned Records (INGRES).
- Hashorganisation (INGRES, UNIFY)
- B*-Baum auf Dense Index (SystemR, Oracle, Adabas D)

CLUSTERING INDEX:

Index über ein oder mehrere Attribute, so daß Tupel mit demselben Attributwert für diese Attribute beieinander gespeichert sind.

Speicherung von Relationen in INGRES:

- Relationen werden als UNIX Dateien gespeichert.
- Datei = Folge von Blöcken mit jeweils 512 Bytes.
- Tupel → Satz → Block

In INGRES bestehen drei Möglichkeiten die Relationen zu organisieren:

1. **Heap**, unsortiert, keine Zugriffsstruktur, Voreinstellung für alle Relationen.

2. **Hash**. Eine Relation kann in Hash-Organisation gebracht werden, indem man folgendes DDL-Statement ausführt:

   `modify R to hash on` $A_1,\ldots,A_k$

   Die Attribute A_1 bis A_k formen hierbei den Schlüssel.

3. **Index**. Eine Organisationsform mit Sparse-Index kann mit dieser DDL-Anweisung erreicht werden:.

   `modify R to isam on` $A_1,\ldots,A_k$

4. Eine vierte Option, die in INGRES möglich ist, ist das Anlegen eines Sekundärindex:

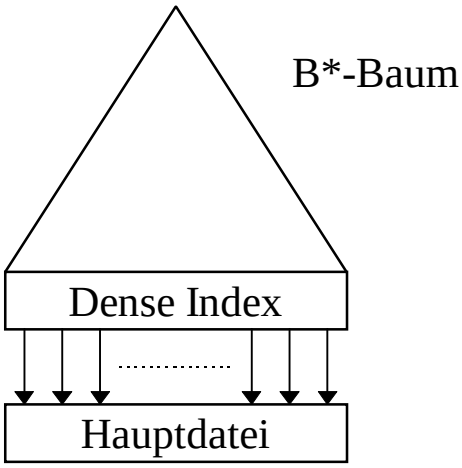
   `index on R is S(`A_1`,…,`A_k`)`

   Die Relation S wird damit zum Sekundärindex über den Attributen A_1,…,A_k. Die Datei für S wird automatisch als ISAM Datei mit Sparse-Index angelegt, dies kann aber durch ein modify Kommando geändert werden.

Speicherung von Relationen in Oracle und ADABAS D

- Relationen werden als Dateien des Dateisystems oder als sogenannte Raw-Partition gespeichert. Letzteres ist ein unformatierter nicht vom Dateisystem verwalteter Abschnitt einer Platte.
- Datei = Folge von Blöcken mit fester Länge,
  in Oracle abhängig vom Betriebssystem,
  in ADABAS D 4 kByte.
- Tupel → Satz → Block

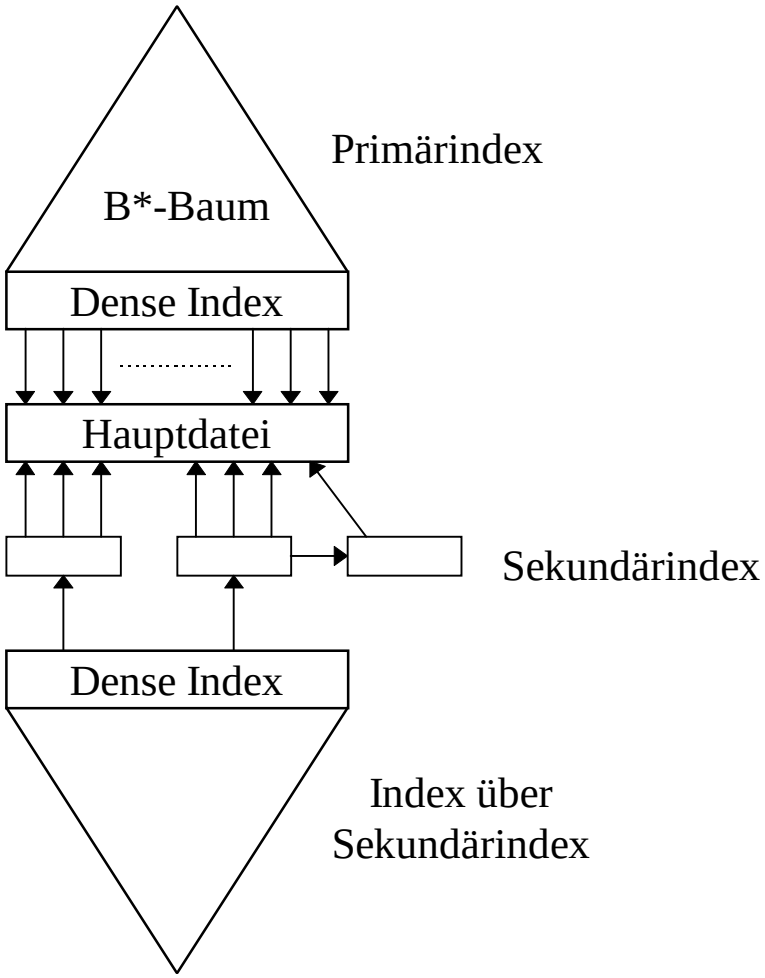
Die Relationen sind grundsätzlich als B*-Baum über einem Dense-Index organisiert.



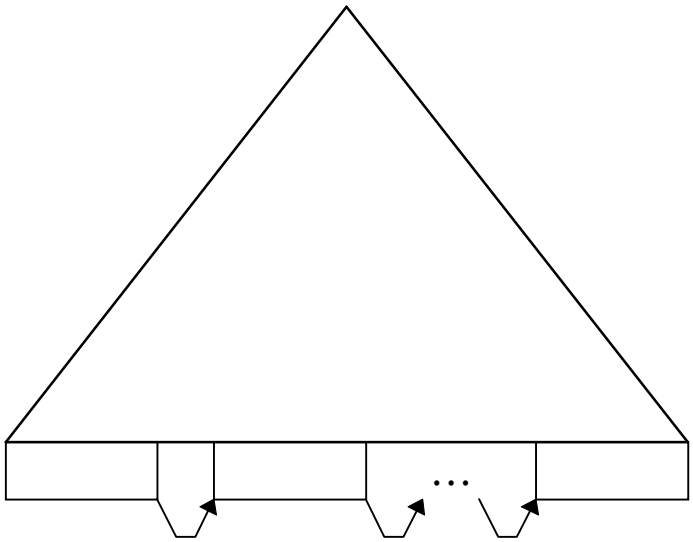
Sekundärindizes werden ebenfalls über B*-Bäume realisiert.

Achtung!: Dies ist nur durch Verwendung eines Dense-Index möglich, da die Sätze der Hauptdatei durch das verwenden mehrere Indizes pinned-down sind.

Datei mit primär- und sekundär-Index:



In ADABAS D werden die Blätter des B*-Baumes miteinander verkettet, um das Durchlaufen der Datei in sequentieller Reihenfolge zu ermöglichen (evtl. auch in Oracle).



CLUSTERING

Abspeichern von Tupeln **verschiedener Relationen** mit gleichen Werten bestimmter Attribute im gleichen Block (**Prejoin**).

Immer ein Satz der einen Relation befindet sich in einem Block mit mehreren Sätzen der anderen Relation.
(Vergleiche: Hierarchisches Datenmodell)

Beispiel: Cluster aus logischer Sicht

Nicht geclusterte Tabellen DEPT & EMP

DEPT

DMPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EMP

EMPNO	ENAME	JOB	MGR	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	800		20
7499	ALLEN	SALESMAN	7698	1600	300.00	30
7521	WARD	SALESMAN	7698	1250	500	30
7566	JONES	MANAGER	7839	3975		20
7654	MARTIN	SALESMAN	7698	1250	1400	30
7782	CLARK	MANAGER	7839	2450		10
7839	KING	PRESIDENT		5000		10

Erzeugen eines DEPT zu EMP Clusters:

```
CREATE CLUSTER DEPT_EMP(DEPTNO NMBER);
```

DEPT Tabelle zum Cluster hinzufügen:

```
ALTER   CLUSTER DEPT_EMP
ADD     TABLE DEPT
WHERE   DEPT.DEPTNO = DEPT_EMP.DEPTNO;
```

EMP Tabelle zum Cluster hinzufügen:

```
ALTER   CLUSTER DEPT_EMP
ADD     TABLE EMP
WHERE   EMP.DEPTNO = DEPT_EMP.DEPTNO;
```

Geclusterte EMP und DEPT Tabelle:

CLUSTER 10

ACCOUNTING		NEWYORK			
7782	CLARK	MANAGER	7839	2450	
7839	KING	PRESIDENT		5000	

CLUSTER 20

RESEARCH		DALLAS			
7369	SMITH	CLERK	7902	800	
7566	JONES	MANAGER	7839	2975	

CLUSTER 30

SALES		CHICAGO			
7499	ALLEN	SALESMAN	7698	1600	300
7521	WARD	SALESMAN	7698	1250	500
7654	MARTIN	SALESMAN	7698	1250	1400

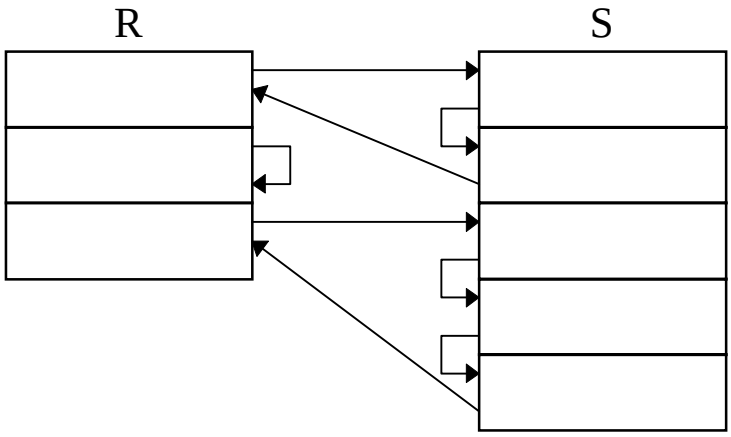
Typischerweise wendet man Clustering an, wenn folgendes gegeben ist:

Zwei Relationen R und S und eine Menge von Attributen X, mit

1. X ist in R und in S
2. X ist Schlüssel von R
3. Wenn $t \in S$, dann gilt $\exists\, t' \in R$ mit:
   $t'[x] = t[x]$.

Clustering ist aus dem Netzwerkmodell entliehen, es wird durch das Set-Konstrukt verwirklicht.

Alternative: Ring Struktur



Vergleich zwischen :

- Cluster R, S und
- Ring Struktur R, S