Modellieren von Hierarchien

Adjazenzliste, Nested Sets, Pfade, Ebenensortierung

Referent: Michael Zimmermann

Zimmermann@SZWeb.de

Access-Stammtisch Rhein-Main

Organisation der Datenbankentwickler
dbdev.org

ts-projects GmbH

Hierarchiearten

- Begriffshierarchie
- Objekthierarchie
  - Finit irreflexiv: Feste Anzahl von Hierarchieebenen verschiedenen Typs
  - Finit reflexiv: Feste Anzahl von Hierarchieebenen gleichen Typs
  - Infinit reflexiv: Die Ebenen können beliebig tief geschachtelt sein

Infinite (offene) Hierarchie

- Abbilden
  - Horizontal (Hierarchische Tabelle)
  - Vertikal (Baum; Treeview oder Liste)
- Modellieren
  - Adjazenzliste
  - Nested Sets
  - Pfad-Methode
  - Ebenensortierung

Grundbegriffe

- Wurzel(n)
- Blätter
- Pfad
- Geschwister
- Vettern
- Elter
- Kinder
- Knoten

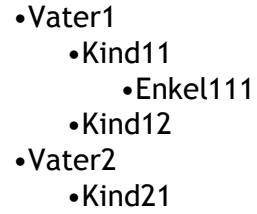
Horizontale Abbildung

- „Hierarchical Flexgrid“
- Darstellung tabellarisch:
  - Spalten
  - Verbundzellen
- Eher unüblich, „Excel-Style“

Ebene0	Ebene1	Ebene2
Vater1	Kind11	Enkel111
	Kind12	
Vater2	Kind21	

Vertikale Abbildung

- Treeview
- Darstellung listenartig:
  - Zeilen
  - Einrückung
- Standard



Adjazenzliste

- Jedes Element bekommt einen Fremdschlüssel auf seinen Elter
- Ur-Wurzel möglich, aber nicht nötig
- Hierarchie in eigene Tabelle auslagern wegen Eingabe-zwang im Fremdschlüssel
- Innerhalb von Geschwister-ebenen beliebige Sortierung möglich

Item	Parent
A	!
AA	A
AB	A
AC	A
ABA	AB
ABB	AB
B	!

In der konkreten Anwendung sollte man zwei Tabellen anlegen: Eine reine Datentabelle und eine reine Hierarchietabelle.

Also nicht

ID; ParentID; Daten…

sondern

ID; Daten ID; ParentID

Die Hierarchietabelle erhält dann zwei Beziehungen auf die ID der Datentabelle: einmal über ID und einmal über ParentID.

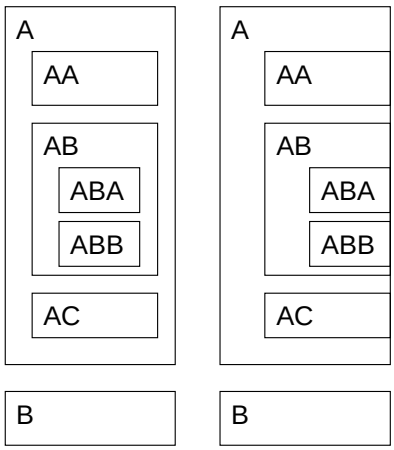
Wurzelelemente erkennt man dann einfach über eine Unterabfrage mit WHERE NOT EXISTS auf das Feld ParentID.

Pfadmethode

- Statt Fremdschlüssel Angabe aller Ahnen
- Hierarchie durch Feldgröße begrenzt
- Nicht atomar; Trennzeichen im Feld
- Indiskutabel

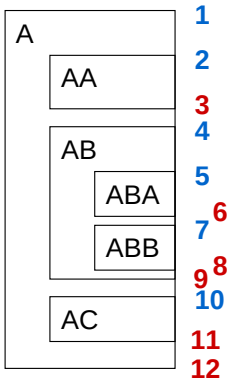
Nested Sets 1

- Teilmengenschachtelung logisch äquivalent zu Baum (fast!)
- A
  - AA
  - AB
    - ABA
    - ABB
  - AC
- B



Nested Sets 2

- Durchnumerieren
- Tabellieren



Item	Left	Right
A	1	12
AA	2	3
AB	4	9
AC	10	11
ABA	5	6
ABB	7	8

Durch die Numerierung ist die Reihenfolge der Elemente komplett festgelegt. Eine freie Sortierung innerhalb der Geschwisterebenen ist nicht möglich.

Eine Nested-Sets-Struktur ist also eine **geordnete** Hierarchie im Gegensatz zur ungeordneten (=sortierbaren) normalen Hierarchie.

Um Elemente einzufügen oder zu löschen, müssen alle in der Reihenfolge hintanstehenden Elemente neu numeriert werden (+2 bzw. -2)!

Dazu sind immer drei Schritte notwendig. Es muß sichergestellt werden, daß diese alle vollständig durchgeführt werden oder widrigenfalls komplett zurückgefahren werden. Also Transaktionen verwenden!

Im Mehrbenutzerbetrieb muß bei jeder Hierarchieänderung die komplette Tabelle gesperrt werden: Änderungen, die logisch nur einen Datensatz betreffen, betreffen physisch viele Datensätze in der Tabelle, da alle nachfolgenden Left- und Right-Werte anzupassen sind.

Beispiele und Code hierzu in der Beispiel-DB.

Geebnete Liste

- Nested Sets ergeben richtige Reihenfolge und Ebene für Baum
- Warum nicht direkt?

Baum	RF	Ebene
A	1	1
AA	2	2
AB	3	2
ABA	4	3
ABB	5	3
AC	6	2
B	7	1

Die geebnete Liste habe ich zur Vereinfachung von Nested Sets eingeführt.

Vorteil: Die Ebene (Generation) ist nicht von anderen Datensätzen abhängig.

Das Anpassen der Reihenfolgenwerte läßt sich durch Lücken umgehen, die ggf. von einer Prozedur alle paar Tage/Monate/Jahre restauriert werden.

Wenn man zum Start statt der Reihenfolgen 1,2,3 z. B. 100, 200, 300 verwendet, können je 99 Elemente eingefügt werden, bevor andere Datensätze geändert werden müssen. Gelegentlich kann dann ein Job außerhalb der Arbeitszeit die ursprünglichen Lücken wiederherstellen.

Bewertung

	Adjazenzliste	Nested Sets	Geebnete Liste
Lesen	Langsam	Schnell	Schnell
Schreiben	Schnell	Langsam	Langsam
Fehleranfällig	Nein	Ja	Nein
Fehlerreichweite	Ein DS	Alle DS	Viele DS
Erforderliche Sperre	Datensatz	Tabelle	Tabelle
Sortierbar	Ja	Nein	Nein
Theoriekonform	Ja	Nein	Bedingt

Parerga zur Baumbefüllung

- Abfragen des ganzen Baumes i. d. R. unnötig
- Treeview-Befüllung beim Klicken
- Problem bei hierarchischen Summen
- Hilfskonstrukt: Summen bei Elementen speichern (getriggert!)
- Motivation für Data-Mining/Hypercubes

In der Beispiel-DB findet man auch ein Beispiel zur sukzessiven Baumbefüllung. Der Geschwindigkeitsnachteil einer Adjazenzliste beim Füllen eines Baumes löst sich nämlich in Wohlgefallen auf, wenn man nicht initial die ganze Hierarchie ausliest, sondern diese beim Klicken auf einen Knoten nur um die jeweils erforderlichen Elemente aufbaut.

Code-Exkurs

Je Modell:

- Abfragen von Standardinformationen
  - Elter/Pfad
  - Anzahl Kinder, Kinder als Menge
  - Ebenennummer
  - Ist Wurzel, ist Blatt?
  - …
- Befüllen eines Treeviews