Import af rekursivt (parent-child) hierarki i Palo

Dette dokument beskriver hvordan et simpelt rekursivt (parent-child) hierarki kan importeres ind i Palo på forskellige måder via SQL og samtidig bibeholde muligheden for at henføre data direkte til overliggende niveauer i hierarkiet. De følgende eksempler kan tilpasses ens eget datasæt.

Først skal vi lave et lille datasæt bestående af en enkelt tabel med et rekursivt (parent-child) hierarki. Tabellen hedder formaal og indeholder produkter, produktgrupper samt det øverste niveau ”Alle produkter”. SQL'en til at danne tabelstruktur og indsætte data ser således ud:

```
DROP TABLE formaal;

CREATE TABLE formaal
(id NUMBER(4),
navn VARCHAR2(30),
parent_id NUMBER (4));

INSERT INTO formaal VALUES
(1, 'Alle produkter', null);

INSERT INTO formaal VALUES
(2, 'Produktgruppe A', 1);

INSERT INTO formaal VALUES
(3, 'Produktgruppe B', 1);

INSERT INTO formaal VALUES
(4, 'Produkt A1', 2);

INSERT INTO formaal VALUES
(5, 'Produkt A2', 2);

INSERT INTO formaal VALUES
(6, 'Produkt A3', 2);

INSERT INTO formaal VALUES
(7, 'Produkt B1', 3);

INSERT INTO formaal VALUES
(8, 'Produkt B2', 3);

COMMIT;
```

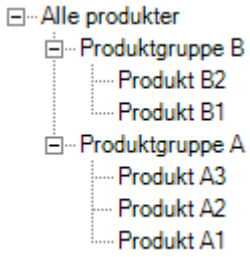
Efter at have dannet tabellen og indsat data vil vi foretage en simpel forespørgsel for at se indholdet af denne:

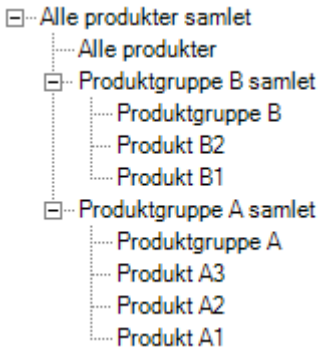
```
SELECT * FROM formaal;
```

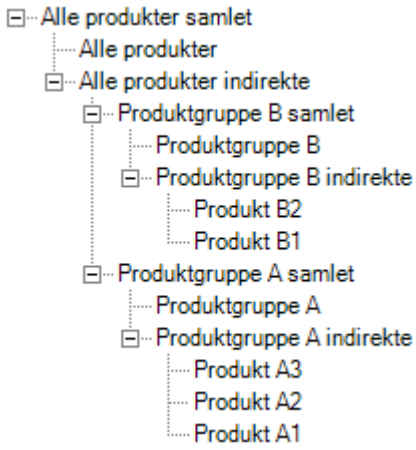
ID	NAVN	PARENT_ID
1	Alle produkter	
2	Produktgruppe A	1
3	Produktgruppe B	1
4	Produkt A1	2
5	Produkt A2	2
6	Produkt A3	2
7	Produkt B1	3
8	Produkt B2	3

Vi vil nu prøve at importere dette datasæt ind i en dimension i Palo på 3 forskellige måder. Kun de 2 sidste giver mulighed for at der kan henføres data direkte til overliggende niveauer i hierarkiet.

Her nedenunder ses de 3 slutresultater, som de vil tage sig ud i Palo. Da jeg ikke udfører nogen eksplicit sortering af data i forespørgslerne er det tilfældigt, at produktgrupper og de enkelte produkter er sorteret i faldende alfabetisk rækkefølge. Man kunne tilføje "`order by kolonnenavn`" til sidst i hver enkelt forespørgsel for at lave en almindelig stigende alfabetisk sortering.







Første udgave er en simpel repræsentation af hierarkiet med "Alle produkter" over produktgrupper som igen er over de enkelte produkter. Da alle elementer som har børn i Palo er konsolideret af data henført til de nederste elementer i hierarkiet er det ikke muligt at henføre data direkte til Produktgrupper og "Alle produkter" i denne situation.

Den anden udgave tilføjer 3 nye elementer til hierarkiet for netop at tage hensyn til, at vi gerne vil kunne henføre data direkte til de overliggende niveauer og samtidig have mulighed for at se hvor meget der samlet set henføres til et element både direkte og indirekte fra underliggende konsoliderede elementer. Konkret optræder hvert oprindelig konsolideret element nu 2 gange – den ene gang ikke-konsolideret (med det originale navn og med henblik på de direkte data) på niveau med de elementer der normalt er børn til elementet, og den anden gang konsolideret (med tilføjelsen "samlet") men nu både med de normale børn samt den direkte version af sig selv.

Denne anden udgave er strukturmæssigt ikke helt elegant, fordi f.eks. "Produktgruppe A" (til direkte henførte data), "Produkt A1", "Produkt A2" og "Produkt A3" optræder på niveau med hinanden, selvom de reelt ikke det, og det er endvidere lidt mere komplekst, hvis man kun vil vise de indirekte data for et konsolideret element.

Derfor vil vi også danne en tredje udgave af hierarkiet, som er den mest komplekse, men tilgengæld også mere korrekt adskiller elementer på forskellige niveauer, og som nemt giver mulighed for kun at se indirekte data til et konsolideret element. I denne tredje udgave optræder alle de oprindelig konsoliderede elementer nu 3 gange i hierarkiet – 1) ikke-konsolideret med det originale navn til direkte henførte data, 2) med tilføjelsen "indirekte" til opsummering af alle underliggende konsoliderede elementer, 3) med tilføjelsen "samlet" til opsummering af de to første.

På de efterfølgende sider vises hvordan SQL-forespørgslerne til dannelsen af hver udgave af hierarkiet skal opbygges på baggrund af det samme originale datasæt.

Jeg tilføjer ikke teksten "direkte" til de elementer, hvor data direkte henføres til, da jeg så også vil skulle tage hensyn til dette, når fakta-data hentes ind i kuberne.

1. Simpelt hierarki i Palo

Til dannelsen af det simple hierarki i Palo anvendes en ”relativ” simpel SQL-forespørgsel, der joiner formaalstabellen med sig selv for at kunne vise navn på både barn og forældre for hver række.

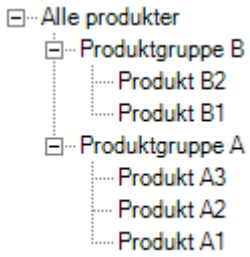
```
SELECT f1.navn child, f2.navn parent
FROM formaal f1, formaal f2
WHERE f1.parent_id = f2.id;
```

CHILD	PARENT
Produktgruppe B	Alle produkter
Produktgruppe A	Alle produkter
Produkt A3	Produktgruppe A
Produkt A2	Produktgruppe A
Produkt A1	Produktgruppe A
Produkt B2	Produktgruppe B
Produkt B1	Produktgruppe B

Ovenstående datasæt importeres ind i Palo vha. følgende PALO.EADD formel, hvor kubens navn er lig ”omkostning”, mens dimensionens navn er ”formål”. Denne formel er den samme for alle udgaver af hierarkiet, da alle SQL-forespørgsler returnere 2 kolonner med henholdsvis barn og forældre.

```
=PALO.EADD("localhost/omkostning";"formål";"n";A1;"B1";1;FALSE)
```

Dette datasæt fører til det simple hierarki i Palo, hvor der ikke kan henføres data direkte til elementer på overliggende niveauer.



Alternativ SQL til at danne ovenstående datasæt

Oracle har en speciel SQL-egenskab (der mig bekendt ikke findes i andre databaser, da det ikke er en standard) til specielt at hente data fra et rekursivt hierarki, som gør at det samme datasæt kan dannes uden at joine tabellen med sig selv, hvilket gør følgende forespørgsel mere effektiv ved store datasæt. I de følgende forespørgsler vil jeg dog fortsætte med den mere almindelige SQL.

```
SELECT navn child, PRIOR navn parent
FROM formaal
CONNECT BY PRIOR id = parent_id
START WITH parent_id IS NULL;
```

2. Halv-avanceret hierarki i Palo

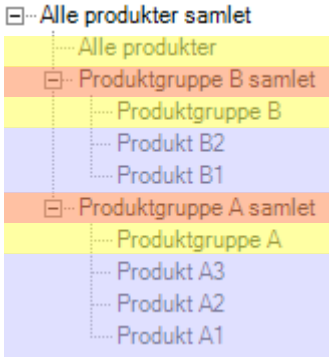
Den anden udgave af hierarkiet kræver flere forespørgsler på kildetabellen for at danne alle data. De enkelte forespørgsler samles til et datasæt vha. ”`UNION ALL`”. Hvis man i stedet for ”`UNION ALL`” bare skrev ”`UNION`” vil databasen frasortere duplikater – dette kræver dog en intern sortering af data og er derfor mere krævende. Derfor bør man altid anvende ”`UNION ALL`”, så længe man ved, at der ikke er duplikater i rækkerne.

Jeg har farvekodet forespørgslen for at vise hvilke rækker, hver enkelt select-sætning returnerer.

Første select finder alle elementer til direkte data, som *ikke* er på laveste niveau (via betingelsen), og sætter forældren lig samme element men med tilføjelsen ”samlet”. Den anden select finder alle elementer, som hverken er på laveste eller højeste niveau og finder deres naturlige forældre i hierarkiet. Både barn og forældre tilføjes ”samlet” i navnet. Endelig finder den tredje og sidste select alle elementer på laveste niveau (via sidste del af betingelsen) og deres tilhørende normale forældre. Kun forældrens navn får her tilføjelsen ”samlet”.

```
SELECT f1.navn child, f1.navn || ' samlet' parent
FROM formaal f1
WHERE f1.id IN (SELECT parent_id FROM formaal)
UNION ALL
SELECT f1.navn || ' samlet' child, f2.navn || ' samlet' parent
FROM formaal f1, formaal f2
WHERE f1.parent_id = f2.id
AND f1.id IN (SELECT parent_id FROM formaal)
UNION ALL
SELECT f1.navn child, f2.navn || ' samlet' parent
FROM formaal f1, formaal f2
WHERE f1.parent_id = f2.id
AND f1.id NOT IN (SELECT parent_id FROM formaal WHERE parent_id IS NOT NULL)
;
```

CHILD	PARENT
Alle produkter	Alle produkter samlet
Produktgruppe A	Produktgruppe A samlet
Produktgruppe B	Produktgruppe B samlet
Produktgruppe B samlet	Alle produkter samlet
Produktgruppe A samlet	Alle produkter samlet
Produkt A3	Produktgruppe A samlet
Produkt A2	Produktgruppe A samlet
Produkt A1	Produktgruppe A samlet
Produkt B2	Produktgruppe B samlet
Produkt B1	Produktgruppe B samlet



3. Avanceret hierarki i Palo

Det tredje og mest komplekse hierarki dannes på samme måde som det forrige, der er nu bare hele 4 select-dele i den samlede forespørgsel. Den grønne select-sætning er den "nye" tilføjelse, men der sker dog også lidt ændringer i de andre select-dele.

Første select finder alle elementer til direkte data, som *ikke* er på laveste niveau (via betingelsen), og sætter forældren lig samme element men med tilføjelsen "samlet". Den anden select er en kopi af den første med den ene forskel at "indirekte" tilføjes børne-elementets navn. Tredje select finder alle elementer, som hverken er på laveste eller højeste niveau og finder deres naturlige forældre i hierarkiet. Teksten "samlet" tilføjes til barnets navn, mens "indirekte" tilføjes til forældren. Endelig finder den fjerde og sidste select alle elementer på laveste niveau (via sidste del af betingelsen) og deres tilhørende normale forældre. Barnets navn forbliver det samme, mens forældren igen får tilføjet "indirekte".

```
SELECT f1.navn child, f1.navn || ' samlet' parent
FROM formaal f1
WHERE f1.id IN (SELECT parent_id FROM formaal)
UNION ALL
SELECT f1.navn || ' indirekte' child, f1.navn || ' samlet' parent
FROM formaal f1
WHERE f1.id IN (SELECT parent_id FROM formaal)
UNION ALL
SELECT f1.navn || ' samlet' child, f2.navn || ' indirekte' parent
FROM formaal f1, formaal f2
WHERE f1.parent_id = f2.id
AND f1.id in (SELECT parent_id FROM formaal)
UNION ALL
SELECT f1.navn child, f2.navn || ' indirekte' parent
FROM formaal f1, formaal f2
WHERE f1.parent_id = f2.id
AND f1.id NOT IN (SELECT parent_id FROM formaal WHERE parent_id IS NOT NULL)
;
```

CHILD	PARENT
Alle produkter	Alle produkter samlet
Produktgruppe A	Produktgruppe A samlet
Produktgruppe B	Produktgruppe B samlet
Alle produkter indirekte	Alle produkter samlet
Produktgruppe A indirekte	Produktgruppe A samlet
Produktgruppe B indirekte	Produktgruppe B samlet
Produktgruppe B samlet	Alle produkter indirekte
Produktgruppe A samlet	Alle produkter indirekte
Produkt A3	Produktgruppe A indirekte
Produkt A2	Produktgruppe A indirekte
Produkt A1	Produktgruppe A indirekte
Produkt B2	Produktgruppe B indirekte
Produkt B1	Produktgruppe B indirekte