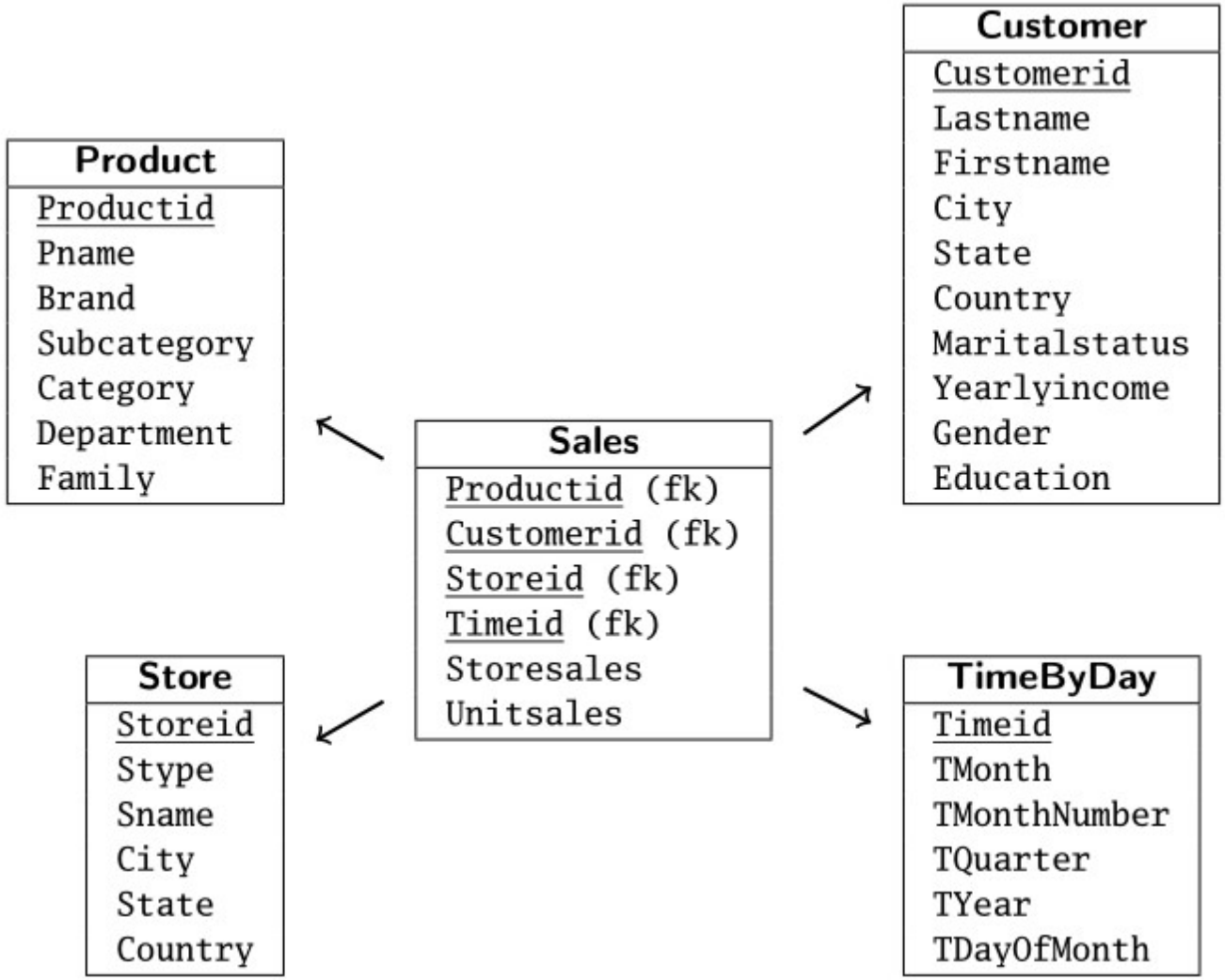
Prof. Dr. Ingo Claßen

Dienstag, 15. März 2022 09:37

- Wiederholung
- Window-Funktionen
- Rekursives SQL


Customer
Customerid
Lastname
Firstname
City
State
Country
Maritalstatus
Yearlyincome
Gender
Education
Product
Productid
Pname
Brand
Subcategory
Category
Department
Family
Sales
Productid (fk)
Customerid (fk)
Storeid (fk)
Timeid (fk)
Storesales
Unitsales
Store
Storeid
Stype
Sname
City
State
Country
TimeByDay
Timeid
TMonth
TMonthNumber
TQuarter
TYear
TDayOfMonth

```
select 'customer' as "table", count(*) as anzahl from customer
union
select 'sales' as "table",  count(*) as anzahl from sales
union
select 'store' as "table",  count(*) as anzahl from store
union
select 'product' as "table",  count(*) as anzahl from product
union
select 'timebyday' as "table",  count(*) as anzahl from timebyday
order by anzahl;
```

	table	COUNT(*)
1	store	25
2	timebyday	730
3	product	1560
4	customer	10281
5	sales	251395

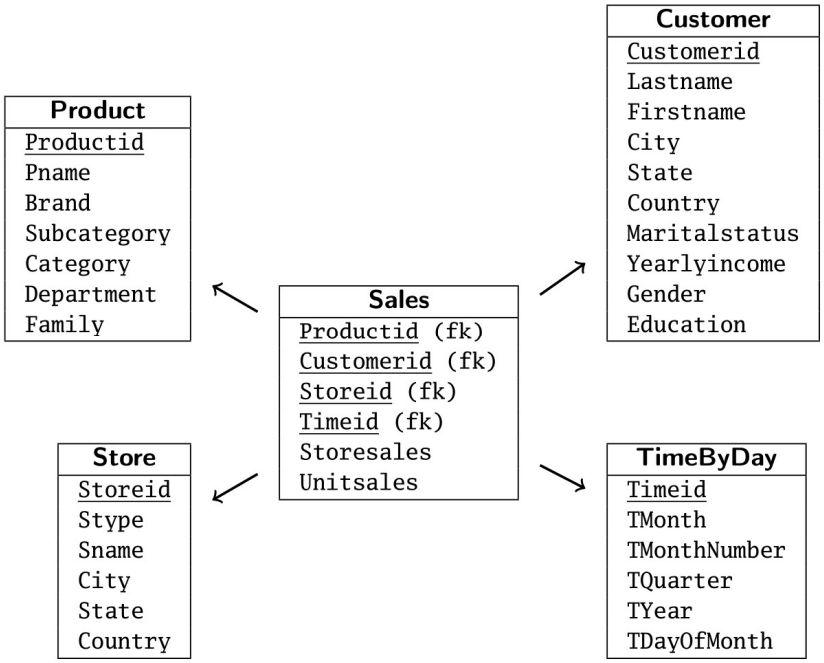
```
-- postgres

select
  schemaname as table_schema,
  relname as table_name,
  n_live_tup as row_count
from pg_stat_user_tables
order by schemaname, n_live_tup desc;
```

Umsatz pro Jahr und Land

Prof. Dr. Ingo Claßen

Mittwoch, 2. Juni 2021 10:23




```
select country, tyear, sum(unitsales)
from sales s
    join timebyday tbd on tbd.timeid=s.timeid
    join customer c on c.customerid=s.customerid
group by country, tyear;
```

	COUNTRY	TYEAR	SUM(UNITSALES)
1	USA	2014	266773
2	Mexico	2015	203914
3	USA	2015	259916
4	Canada	2015	46157

Eindimensionale Darstellung

Zweidimensionale Darstellung (z.B. als Pivotabelle)

	2014	2015
Canada		46157
Mexico		203914
USA	266773	259916

Analyse Produkthierarchie 1

Prof. Dr. Ingo Claßen

Mittwoch, 28. April 2021 09:32

```
select family, count(*)
from product
group by family
order by family;
```

	FAMILY	COUNT(*)
1	Drink	145
2	Food	1120
3	Non-Consumable	295

```
select department, count(*)
from product
group by department
order by department;
```

	DEPARTMENT	COUNT(*)
1	Alcoholic Beverages	40
2	Baked Goods	45
3	Baking Goods	120
4	Beverages	80
5	Breakfast Foods	20
6	Canned Foods	110
7	Canned Products	10
8	Carousel	5
9	Checkout	10
10	Dairy	100
11	Deli	70
12	Eggs	25
13	Frozen Foods	155
14	Health and Hygiene	95
15	Household	160
16	Meat	10
17	Periodicals	25
18	Produce	220
19	Seafood	10
20	Snack Foods	180
21	Snacks	40
22	Starchy Foods	30

```
select family, department, count(*)
from product
group by family, department
order by family, department;
```

	FAMILY	DEPARTMENT	COUNT(*)
1	Drink	Alcoholic Beverages	40
2	Drink	Beverages	80
3	Drink	Dairy	25
4	Food	Baked Goods	45
5	Food	Baking Goods	120
6	Food	Breakfast Foods	20
7	Food	Canned Foods	110
8	Food	Canned Products	10
9	Food	Dairy	75
10	Food	Deli	70
11	Food	Eggs	25
12	Food	Frozen Foods	155
13	Food	Meat	10
14	Food	Produce	220
15	Food	Seafood	10
16	Food	Snack Foods	180
17	Food	Snacks	40
18	Food	Starchy Foods	30
19	Non-Consumable	Carousel	5
20	Non-Consumable	Checkout	10
21	Non-Consumable	Health and Hygiene	95
22	Non-Consumable	Household	160
23	Non-Consumable	Periodicals	25

Ups, Datenfehler, keine echte Hierarchie

```
select department
from
  (select family, department
  from product
  group by family, department)
group by department
having count(*)>1;
```

	DEPARTMENT
1	Dairy

Einfluss Ausbildungsstufe auf Einkommen

```
select education,
  case
    when yearlyincome in ('$10K - $30K', '$30K - $50K', '$50K - $70K') then 'L1'
    else 'L2'
  end as annual_income, count(*)
from customer
group by
  education,
  case
    when yearlyincome in ('$10K - $30K', '$30K - $50K', '$50K - $70K') then 'L1'
    else 'L2'
  end
order by
  education,
  annual_income;
```

	EDUCATION	ANNUAL_INCOME	COUNT(*)
1	Bachelors Degree	L1	1583
2	Bachelors Degree	L2	1036
3	Graduate Degree	L1	38
4	Graduate Degree	L2	501
5	High School Degree	L1	2327
6	High School Degree	L2	712
7	Partial College	L1	847
8	Partial College	L2	143
9	Partial High School	L1	2599
10	Partial High School	L2	495

```
with p3top as (
  select productid, sum(unitsales) as sales2014
  from sales s join timebyday t on s.timeid=t.timeid
  where tyear=2014
  group by productid
  order by sum(unitsales) desc fetch first 3 rows only
)
select
  s.productid, sales2014, sum(unitsales) as sales2015,
  sum(unitsales) - sales2014 as diff
from sales s join timebyday t on s.timeid=t.timeid
  join p3top on s.productid=p3top.productid
where tyear=2015
group by s.productid, sales2014;
```

	PRODUCTID	SALES2014	SALES2015	DIFF
1	952	267	337	70
2	1452	257	282	25
3	549	258	335	77

Funktionsaufrufe in Ausgabespalten:

sum(valcol) **over**()	Gesamtsumme
sum(valcol) **over**(**partition by** groupcol)	Teilsumme pro Gruppe
sum(valcol) **over**(**order by** ordercol)	Kumulierte Summe
sum(valcol) **over**(**partition by** groupcol **order by** ordercol)	Kumulierte Summe pro Gruppe
rank() **over**(**partition by order by** ordercol)	Rangbildung (für Top-N-Analysen)

	TYEAR	TMONTHNUMBER	FAMILY	UNITSALES	TOTAL_MONTH	PERCENT_OF_MONTH	TOTAL	PERCENT_OF_TOTAL
1	2014	1	Drink	1910	21628	8,8	776760	0,2
2	2014	1	Food	15604	21628	72,1	776760	2
3	2014	1	Non-Consumable	4114	21628	19	776760	0,5
4	2014	2	Drink	1951	20957	9,3	776760	0,3
5	2014	2	Food	15142	20957	72,3	776760	1,9
6	2014	2	Non-Consumable	3864	20957	18,4	776760	0,5
7	2014	3	Drink	2115	23706	8,9	776760	0,3
8	2014	3	Food	17063	23706	72	776760	2,2
9	2014	3	Non-Consumable	4528	23706	19,1	776760	0,6
10	2014	4	Drink	1948	20179	9,7	776760	0,3
11	2014	4	Food	14393	20179	71,3	776760	1,9
12	2014	4	Non-Consumable	3838	20179	19	776760	0,5
13	2014	5	Drink	2039	21081	9,7	776760	0,3
14	2014	5	Food	15055	21081	71,4	776760	1,9
15	2014	5	Non-Consumable	3987	21081	18,9	776760	0,5
16	2014	6	Drink	1908	21350	8,9	776760	0,2
17	2014	6	Food	15377	21350	72	776760	2
18	2014	6	Non-Consumable	4065	21350	19	776760	0,5
19	2014	7	Drink	2205	23763	9,3	776760	0,3
20	2014	7	Food	17036	23763	71,7	776760	2,2
21	2014	7	Non-Consumable	4522	23763	19	776760	0,6
22	2014	8	Drink	1921	21697	8,9	776760	0,2

```
with
  basecube as (
    select
      family, tyear, tmonthnumber,
      sum(unitsales) as unitsales
    from sales s
        join timebyday tbd on tbd.timeid=s.timeid
        join product p on p.productid=s.productid
    group by tyear, tmonthnumber, family
)
select
  tyear, tmonthnumber, family,
  unitsales,
  sum(unitsales) over(partition by tyear, tmonthnumber) as total_month,
  round(unitsales/sum(unitsales) over(partition by tyear, tmonthnumber), 3) * 100 as percent_of_month,
  sum(unitsales) over() as total,
  round(unitsales/sum(unitsales) over(), 3) * 100 as percent_of_total
from basecube
order by tyear, tmonthnumber, family;
```

```
with
  basecube as (
    select
      tyear, tmonthnumber,
      sum(unitsales) as unitsales
    from sales s
        join timebyday tbd on tbd.timeid=s.timeid
        join product p on p.productid=s.productid
    group by tyear, tmonthnumber
)
select
  tmonthnumber, unitsales,
  sum(unitsales) over(order by tmonthnumber) as cumulative
from basecube
where tyear=2014
order by tmonthnumber;
```

	TMONTHNUMBER	UNITSALES	CUMULATIVE
1	1	21628	21628
2	2	20957	42585
3	3	23706	66291
4	4	20179	86470
5	5	21081	107551
6	6	21350	128901
7	7	23763	152664
8	8	21697	174361
9	9	20388	194749
10	10	19958	214707
11	11	25270	239977
12	12	26796	266773

```
with
  basecube as (
    select
      family, tyear, tmonthnumber,
      sum(unitsales) as unitsales
    from sales s
        join timebyday tbd on tbd.timeid=s.timeid
        join product p on p.productid=s.productid
    group by family, tyear, tmonthnumber
)
select
  family, tmonthnumber, unitsales,
  sum(unitsales) over(partition by family order by tmonthnumber) as cumulative
from basecube
where tyear=2014
order by family, tmonthnumber;
```

	FAMILY	TMONTHNUMBER	UNITSALES	CUMULATIVE
1	Drink	1	1910	1910
2	Drink	2	1951	3861
3	Drink	3	2115	5976
4	Drink	4	1948	7924
5	Drink	5	2039	9963
6	Drink	6	1908	11871
7	Drink	7	2205	14076
8	Drink	8	1921	15997
9	Drink	9	1939	17936
10	Drink	10	1898	19834
11	Drink	11	2344	22178
12	Drink	12	2419	24597
13	Food	1	15604	15604
14	Food	2	15142	30746
15	Food	3	17063	47809
16	Food	4	14393	62202
17	Food	5	15055	77257
18	Food	6	15377	92634
19	Food	7	17036	109670
20	Food	8	15741	125411

Top 10 Produkte

Prof. Dr. Ingo Claßen

Donnerstag, 3. Juni 2021 09:33

```
with
  basecube as (
    select
      pname,
      sum(unitsales) as unitsales
    from sales s
        join product p on p.productid=s.productid
    group by pname
)
select
  rank() over(order by unitsales desc) as ranklevel,
  pname,
  unitsales
from basecube
order by ranklevel
fetch first 10 rows only;
```

	RANKLEVEL	PNAME	UNITSALES
1	1	Tell Tale Fresh Lima Beans	645
2	2	Steady Whitening Toothpast	634
3	3	Ebony Mixed Nuts	629
4	4	Great English Muffins	622
5	4	Hilltop Mint Mouthwash	622
6	6	Steady Childrens Cold Remedy	621
7	7	Hermanos Green Pepper	614
8	8	Moms Roasted Chicken	613
9	9	Fabulous Apple Juice	612
10	9	Nationeel Golden Raisins	612

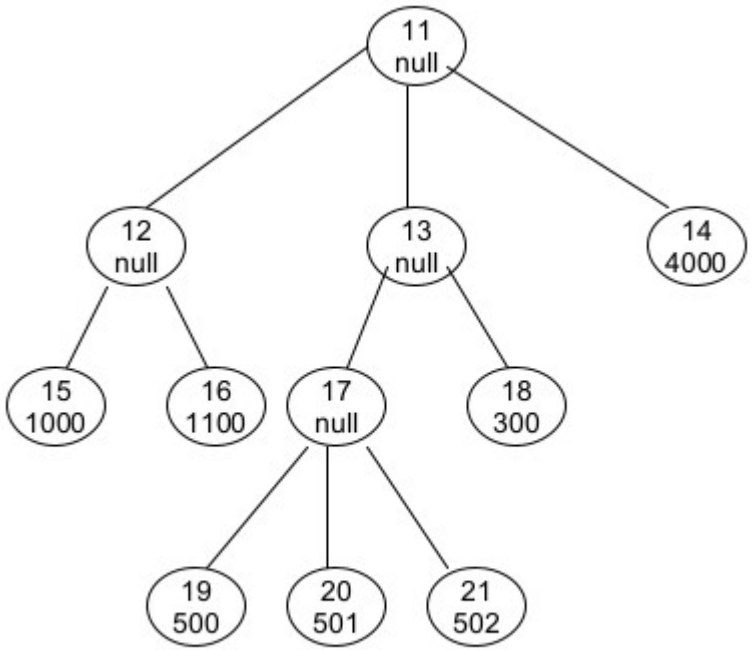
```
with
 basecube as (
  select
   category,
   sum(unitsales) as unitsales
  from sales s
     join product p on p.productid=s.productid
  group by category
 ),
 revenues as (
  select
   category,
   unitsales,
   sum(unitsales) over(order by unitsales desc) as cum_revenue,
   sum(unitsales) over() as total_revenue
  from basecube
 )
select
 category, unitsales, cum_revenue, total_revenue,
 case
  when cum_revenue < 0.5 * total_revenue then 'A'
  when cum_revenue < 0.7 * total_revenue then 'B'
  else 'C'
 end as cat
from revenues
order by unitsales desc;
```

	CATEGORY	UNITSALES	CUM_REVENUE	TOTAL_REVENUE	CAT
1	Vegetables	94626	94626	776760	A
2	Snack Foods	89179	183805	776760	A
3	Dairy	49492	233297	776760	A
4	Meat	39990	273287	776760	A
5	Fruit	39798	313085	776760	A
6	Jams and Jellies	34453	347538	776760	A
7	Breakfast Foods	24870	372408	776760	A
8	Baking Goods	24735	397143	776760	B
9	Bread	22852	419995	776760	B
10	Canned Soup	22769	442764	776760	B
11	Candy	20096	462860	776760	B
12	Bathroom Products	20054	482914	776760	B
13	Electrical	20000	502914	776760	B
14	Beer and Wine	19907	522821	776760	B
15	Paper Products	19821	542642	776760	B
16	Frozen Desserts	17634	560276	776760	C
17	Specialty	15103	575379	776760	C
18	Starchy Foods	15076	590455	776760	C
19	Kitchen Products	12434	602889	776760	C
20	Magazines	12400	615289	776760	C

```
with recursive t(n) as (
    values (1)
  union all
    select n+1 from t where n < 100
)
select sum(n) from t;
```

```
create table tree0 (
  id integer not null primary key,
  p integer,
  v integer
);
```

```
insert into tree0 values
  (11, null, null),
  (12, 11, null),
  (13, 11, null),
  (14, 11, 4000),
  (15, 12, 1000),
  (16, 12, 1100),
  (17, 13, null),
  (18, 13, 300),
  (19, 17, 500),
  (20, 17, 501),
  (21, 17, 502)
;
```



Mittwoch, 9. März 2022 09:22

```
with recursive
  tree1(id, p, v, lvl, pth) as (
      select id, p, v, 1, id::varchar(200)
      from tree0
      where p is null
    union all
      select t0.id, t0.p, t0.v, lvl+1, (pth || '/' || t0.id)::varchar(200)
      from tree0 t0 join tree1 t1 on t0.p=t1.id
  ),
  tree2(id, p, v, lvl, pth, kind) as (
    select
      id, p, v, lvl, pth,
      case
        when lvl=(select min(lvl) from tree1) then 'root'
        when (not exists (select * from tree1 below where below.p =tree1.id)) then 'leaf'
        else 'inner'
      end
    from tree1
  )
select id, p, v, lvl, pth, kind
from tree2
order by lvl;
```

123 id	123 p	123 v	123 lvl	ABC pth	ABC kind
11	[NULL]	[NULL]	1	11	root
12	11	[NULL]	2	11/12	inner
13	11	[NULL]	2	11/13	inner
14	11	4.000	2	11/14	leaf
15	12	1.000	3	11/12/15	leaf
16	12	1.100	3	11/12/16	leaf
17	13	[NULL]	3	11/13/17	inner
18	13	300	3	11/13/18	leaf
19	17	500	4	11/13/17/19	leaf
20	17	501	4	11/13/17/20	leaf
21	17	502	4	11/13/17/21	leaf

```
with recursive
  tree1(id, p, v, pth) as (
      select id, p, v, id::varchar(200)
      from tree0
      where not exists (select * from tree0 below where below.p =tree0.id)
    union all
      select t0.id, t0.p, t1.v, (pth || '/' || t0.id)::varchar(200)
      from tree0 t0 join tree1 t1 on t1.p=t0.id
  )
select id, p, v, pth
from tree1
order by id;
```

123 id	123 p	123 v	ABC pth
11	[NULL]	1.000	15/12/11
11	[NULL]	502	21/17/13/11
11	[NULL]	501	20/17/13/11
11	[NULL]	500	19/17/13/11
11	[NULL]	4.000	14/11
11	[NULL]	300	18/13/11
11	[NULL]	1.100	16/12/11
12	11	1.000	15/12
12	11	1.100	16/12
13	11	300	18/13
13	11	502	21/17/13
13	11	501	20/17/13
13	11	500	19/17/13
14	11	4.000	14
15	12	1.000	15
16	12	1.100	16
17	13	500	19/17
17	13	502	21/17
17	13	501	20/17
18	13	300	18
19	17	500	19
20	17	501	20
21	17	502	21

```
with recursive
  tree1(id, p, v) as (
      select id, p, v
      from tree0
      where not exists (
        select *
        from tree0 below
        where below.p =tree0.id
      )
    union all
      select t0.id, t0.p, t1.v
      from tree0 t0 join tree1 t1 on t1.p=t0.id
  )
select id, sum(v)
from tree1
group by id
order by id;
```

123 id	123 sum
11	7.903
12	2.100
13	1.803
14	4.000
15	1.000
16	1.100
17	1.503
18	300
19	500
20	501
21	502