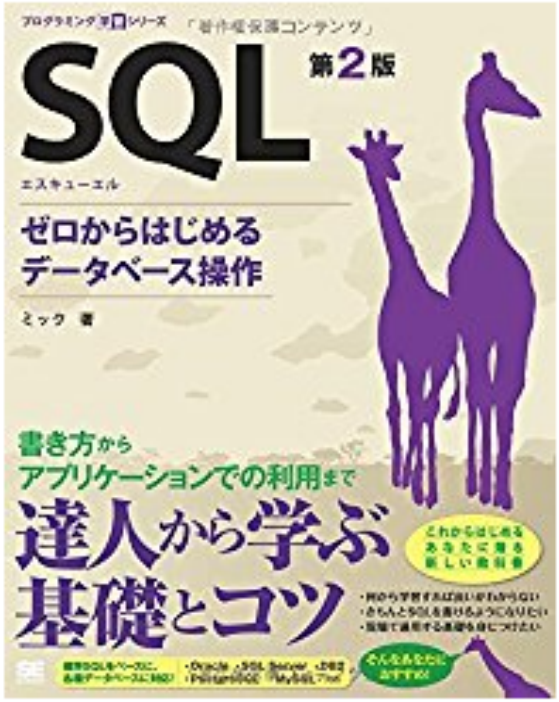


『おうちで学べるデータベースのきほん』

初級

データベースに初めてふれる学生や新入社員の方々に向けた入門書。木村明治さんとの共著。

サポートページ

『SQL 第2版 ゼロからはじめるデータベース操作』

初級

SQLを初めて学習するの方々に向けた入門書。おかげさまで第二版。

サポートページ

『SQL実践入門』

中～上級

大量データを扱う際にハイパフォーマンスを実現するためのSQLの書き方と、実行計画を見ながら性能を解析する技術がテーマです。

サポートページ

『達人に学ぶ SQL徹底指南書』

初～中級

CodeZineの連載記事を中心にSQLの高度なテクニックとその原理の解説です。入門を終えて中級へ進もうとする方々（でもまだセルコには歯が立たない）のサブテキストとして。

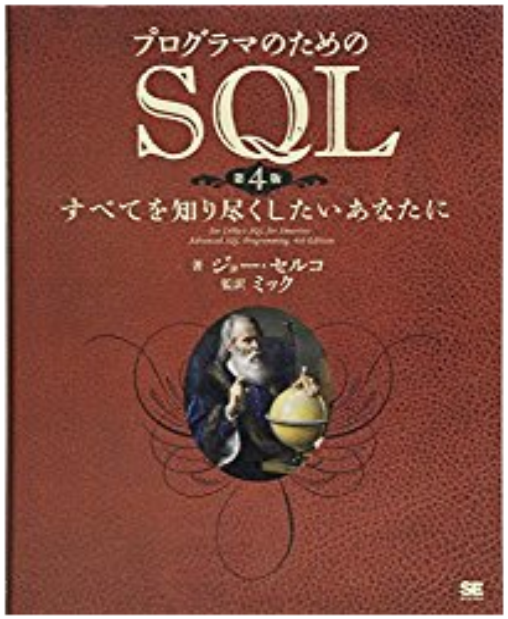
サポートページ

『達人に学ぶ DB設計徹底指南書』

初～中級

データベースの論理設計（ER、正規化）、物理設計（RAID、アーキテクチャ）などについての解説です。

サポートページ

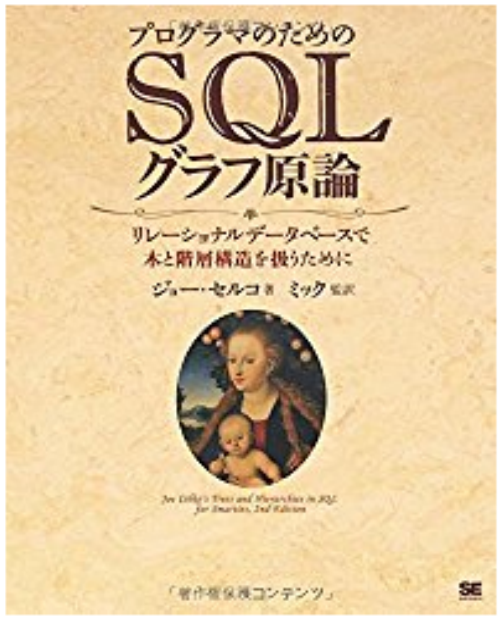


『プログラマのためのSQL 第4版』

上級

Joe Celko著。世界で一番読まれているであろうSQLの解説書。SQLを1年以上使っているDBエンジニアを対象にした上級テクニックを解説。

サポートページ



『プログラマのためのSQL グラフ原論』

上級

Joe Celko著。『プログラマのためのSQL』のスピンオフで、木と階層構造を扱う方法論のみにフォーカスしたマニア度の高い一冊。

サポートページ

『SQLパズル 第2版』

上級

Joe Celko著。SQLの練習問題。難易度は高い。もし難しいと思ったら上掲の拙著『SQL徹底指南書』へ。

サポートページ

PostgreSQLでSQLパズルの問題を解く

概要

PostgreSQL Conference Japan 2017で講演させていただくことになりまして、
そのまとめページです。

PostgreSQLでSQLパズル(2版)の問題を解いて、
私のSQLの思考法と脳内のイメージを解説します。

PostgreSQL9.3で動作確認してます。

プログラム

17. 人材紹介会社

Skillsテーブル

ID	Skill
100	会計
100	在庫管理
100	製造
200	会計
200	在庫管理
300	製造
400	在庫管理
400	製造
500	会計
500	製造

下記の条件を満たすIDを出力する。

Skill = '製造' のレコードが存在し、かつ、Skill = '在庫管理' のレコードが存在する。
または
Skill = '会計' のレコードが存在する。

出力結果

ID
100
200
400
500

```
-- ■■■ データ作成スクリプト ■■■
create table Skills(
ID    integer,
Skill text,
primary key(ID,Skill));

insert into Skills values(100,'会計'),
                         (100,'在庫管理'),
                         (100,'製造'),
                         (200,'会計'),
                         (200,'在庫管理'),
                         (300,'製造'),
                         (400,'在庫管理'),
                         (400,'製造'),
                         (500,'会計'),
                         (500,'製造');
```

```
-- ■■■ 解1 Case式とSum関数を組み合わせる方法 ■■■
select ID
  from Skills
group by ID
having sum(case Skill when '製造'     then 1 else 0 end) > 0
   and sum(case Skill when '在庫管理' then 1 else 0 end) > 0
    or sum(case Skill when '会計'     then 1 else 0 end) > 0
order by ID;
```

```
-- ■■■ 解2 Bool_Or関数を使う方法 ■■■
select ID
  from Skills
group by ID
having Bool_Or(Skill = '製造') and Bool_Or(Skill = '在庫管理')
    or Bool_Or(Skill = '会計')
order by ID;
```

PostgreSQLには、
存在肯定命題をチェックする用のBool_Or関数
全称肯定命題をチェックする用のBool_And関数があります。

SQLのイメージは下記となります。group by IDで赤線を引いてます。

	ID integer	Skill text
1	100	会計
2	100	在庫管理
3	100	製造
4	200	会計
5	200	在庫管理
6	300	製造
7	400	在庫管理
8	400	製造
9	500	会計
10	500	製造

18. ダイレクトメール

Consumersテーブル

ConName	Address	Con_ID	Fam	
Bob	AAAA	1	null	←削除対象行
Joe	BBBB	3	null	←削除対象行
Mark	CCCC	5	null	
Mary	AAAA	2	1	
Vickie	BBBB	4	3	
Wayne	DDDD	6	null	

Fam列がnullで、かつ、Address列が同じ行が存在したら、削除する。

```
-- ■■■ データ作成スクリプト ■■■
create table Consumers(
ConName text,
Address text,
Con_ID  integer,
Fam     integer);

insert into Consumers values('Bob'   ,'AAAA',1,null),
                             ('Joe'   ,'BBBB',3,null),
                             ('Mark'  ,'CCCC',5,null),
                             ('Mary'  ,'AAAA',2,   1),
                             ('Vickie','BBBB',4,   3),
                             ('Wayne' ,'DDDD',6,null);
```

```
-- ■■■ 解1 削除可能なビュー(DeletableView)もどきを使う方法 ■■■
with DelView as(
select ConName
  from (select ConName,Fam,count(*) over(partition by Address) as Cnt
          from Consumers) tmp
 where Fam is null
   and Cnt > 1)
delete from Consumers
 where ConName in(select ConName from DelView);
```

with句で定義したビューに対してDelete文を実行すると
PostgreSQL9.3では文法エラーになりますので、In述語で定義したビューを参照してます。

```
-- ■■■ 解2 サブクエリを使う方法 ■■■
delete from Consumers
 where Fam is null
   and Address in(select Address
                    from Consumers
                   group by Address
                   having count(*) > 1);
```

サブクエリを使う方法もあります。
SQLのイメージは下記となります。group by Addressで赤線を引いてます。

	ConName text	Address text	Con_ID integer	Fam integer
1	Bob	AAAA	1	
2	Mary	AAAA	2	1
3	Joe	BBBB	3	
4	Vickie	BBBB	4	3
5	Mark	CCCC	5	
6	Wayne	DDDD	6	

21. 飛行機と飛行士

PilotSkillsテーブル

Pilot	Plane
Celko	ペリカン
Higgins	ハンター
Higgins	ファルコ
Higgins	ペリカン
Jones	ハンター
Jones	ファルコ
Smith	ジャビィ
Smith	ハンター
Smith	ファルコ
Wilson	ジャビィ
Wilson	ハンター
Wilson	ファルコ
Wilson	ラップル

Hangarテーブル

Plane
ジャビィ
ハンター
ファルコ

Hangarテーブルの全レコードのPlaneを持つPilotを出力する。
(Hangarテーブルが空集合のケースは考えません)

出力結果

Pilot
Smith
Wilson

```
-- ■■■ データ作成スクリプト ■■■
create table PilotSkills(
Pilot text,
Plane text,
primary key(Pilot, Plane));

insert into PilotSkills values('Celko'   ,'ペリカン'),
                               ('Higgins','ハンター'),
                               ('Higgins','ファルコ'),
                               ('Higgins','ペリカン'),
                               ('Jones'  ,'ハンター'),
                               ('Jones'  ,'ファルコ'),
                               ('Smith'  ,'ジャビィ'),
                               ('Smith'  ,'ハンター'),
                               ('Smith'  ,'ファルコ'),
                               ('Wilson' ,'ジャビィ'),
                               ('Wilson' ,'ハンター'),
                               ('Wilson' ,'ファルコ'),
                               ('Wilson' ,'ラップル');

create table Hangar(Plane text primary key);
insert into Hangar values('ジャビィ'),
                          ('ハンター'),
                          ('ファルコ');
```

```
-- ■■■ 解1 except集合演算を使う方法 ■■■
select distinct Pilot
  from PilotSkills a
 where not exists(select b.Plane from Hangar b
                    except all
                    select c.Plane from PilotSkills c
                     where c.Pilot = a.Pilot);
```

except集合演算で求めた差集合が、空集合かで
包含関係をチェックしてます。

```
-- ■■■ 解2 Window関数のcount関数で件数を求めてから結合する方法 ■■■
select b.Pilot
from (select Plane,count(*) over() as cnt
        from Hangar) a
Join PilotSkills b
  on a.Plane = b.Plane
group by b.Pilot,a.cnt
having count(*) = a.cnt;
```

最初にインラインビューでWindow関数のcount関数を使って、
Hangarテーブルの行数を列別名cntとして求めてます。
次に、Planeが等しいことを条件として内部結合(等価結合)してます。
そして、group by b.Pilot,a.cntでグループ化し、having count(*) = a.cntによって、
内部結合した結果の件数がHangarテーブルの行数と同じであることを抽出条件としてます。

SQLのイメージは下記となります。
Window関数のcount関数に対応する黄緑線を引き、
Planeが等しいことを条件とした内部結合(等価結合)に対応するベン図をイメージしながら青線や紫線などを引き、
group by b.Pilot,a.cntに対応する赤線を引いてます。

	Plane text	cnt bigint
1	ジャビィ	3
2	ハンター	3
3	ファルコ	3

	Pilot text	Plane text
1	Celko	ペリカン
2	Higgins	ハンター
3	Higgins	ファルコ
4	Higgins	ペリカン
5	Jones	ハンター
6	Jones	ファルコ
7	Smith	ジャビィ
8	Smith	ハンター
9	Smith	ファルコ
10	Wilson	ジャビィ
11	Wilson	ハンター
12	Wilson	ファルコ
13	Wilson	ラップル

```
-- ■■■ 解3 array_agg関数を使う方法1 ■■■
select Pilot
  from PilotSkills a
group by Pilot
having exists(select 1 from Hangar b
                having array_agg(a.Plane) @> array_agg(b.Plane));
```

array_agg関数は、配列型を返しますので、集合同士の包含関係を調べることができます。

```
-- ■■■ 解4 array_agg関数を使う方法2 ■■■
select Pilot
  from PilotSkills a
group by Pilot
having (select array_agg(a.Plane) @> array_agg(b.Plane)
          from Hangar b);
```

Boolean型を返すスカラーサブクエリをhaving句で使ってもいいです。

SQLのイメージは下記となります。group by Pilotで赤線を引いて、
array_agg関数で、ベン図をイメージしてます。

	Pilot text	Plane text
1	Celko	ペリカン
2	Higgins	ハンター
3	Higgins	ファルコ
4	Higgins	ペリカン
5	Jones	ハンター
6	Jones	ファルコ
7	Smith	ジャビィ
8	Smith	ハンター
9	Smith	ファルコ
10	Wilson	ジャビィ
11	Wilson	ハンター
12	Wilson	ファルコ
13	Wilson	ラップル

27. 等しい集合を見つける

達人に学ぶ SQL徹底指南書の、等しい部分集合を見つける(132ページ)で扱われている、
同じ問題を解きます。

SupPartsテーブル

sup	part
A	ボルト
A	ナット
A	パイプ
B	ボルト
B	パイプ
C	ボルト
C	ナット
C	パイプ
D	ボルト
D	パイプ
E	ヒューズ
E	ナット
E	パイプ
F	ヒューズ

数も種類もまったく同じpartを取り扱うsupの組み合わせを求めます。

出力結果

s1	s2
A	C
B	D

```
-- ■■■ データ作成スクリプト ■■■
create table SupParts(
sup  text,
part text,
primary key(sup,part));

insert into SupParts values('A','ボルト'),
                            ('A','ナット'),
                            ('A','パイプ'),
                            ('B','ボルト'),
                            ('B','パイプ'),
                            ('C','ボルト'),
                            ('C','ナット'),
                            ('C','パイプ'),
                            ('D','ボルト'),
                            ('D','パイプ'),
                            ('E','ヒューズ'),
                            ('E','ナット'),
                            ('E','パイプ'),
                            ('F','ヒューズ');
```

```
-- ■■■ 解1 内部結合後の件数を調べる方法 ■■■
with tmp as(
select sup,part,count(*) over(partition by sup) as cnt
  from SupParts)
select a.sup as s1,b.sup as s2
  from tmp a Join tmp b
    on a.sup < b.sup
   and a.cnt = b.cnt
   and a.part = b.part
group by a.sup,b.sup,a.cnt
having count(*) = a.cnt
order by a.sup,b.sup;
```

まず、Window関数のcount関数でsupごとの件数を求めた結果を、仮想表tmpとしてます。

次に、supが自分より大きいこと、件数が等しいこと、partが等しいこと
を条件として自己内部結合させてます。
with句は、select文の結果同士を自己結合させる際に使うと便利です。

そして、GroupBy句でsupの組み合わせでグループ化して、
having count(*) = a.cntで、内部結合によって件数が減らなかったsupの組み合わせを出力対象としてます。

SQLのイメージは、下記となります。
仮想表tmpのselect文のcount(*) over(partition by sup)に対応する赤線と黄緑線を引いてから、
仮想表tmp同士の、supが自分より大きいこと、件数が等しいこと、partが等しいこと
を条件とした自己内部結合をイメージしてます。

tmp a

	sup text	part text	cnt bigint
1	A	ナット	3
2	A	パイプ	3
3	A	ボルト	3
4	B	パイプ	2
5	B	ボルト	2
6	C	ナット	3
7	C	パイプ	3
8	C	ボルト	3
9	D	パイプ	2
10	D	ボルト	2
11	E	ナット	3
12	E	パイプ	3
13	E	ヒューズ	3
14	F	ヒューズ	1

tmp b

	sup text	part text	cnt bigint
1	A	ナット	3
2	A	パイプ	3
3	A	ボルト	3
4	B	パイプ	2
5	B	ボルト	2
6	C	ナット	3
7	C	パイプ	3
8	C	ボルト	3
9	D	パイプ	2
10	D	ボルト	2
11	E	ナット	3
12	E	パイプ	3
13	E	ヒューズ	3
14	F	ヒューズ	1

数学の集合では、集合の相等性を調べる公式として、以下が有名ですが、
(A ⊆ B) かつ (A ⊇ B) ⇔ (A = B)

(集合Aと集合Bの要素数が等しい) かつ (A ⊆ B) ⇔ (A = B) も成立します。
集合Aと集合Bが両方とも空集合の場合は、自明ですし、
集合Aと集合Bが両方とも空集合でない場合は、要素数が等しくて包含関係が成立するのは、A=Bの場合しかないからです。

要素数は、Window関数のcount関数を使えば求まりますし、
包含関係は、要素が等しいことを条件として内部結合して、要素数が減らなかったかを調べれば分かります。

```
-- ■■■ 解2 配列型を使う方法1 ■■■
with tmp as(
select sup,array_agg(part) as ArrPart
  from SupParts
group by sup)
select a.sup,b.sup
  from tmp a Join tmp b
    on a.sup < b.sup
   and array_length(a.ArrPart,1) = array_length(b.ArrPart,1)
   and a.ArrPart <@ b.ArrPart
order by a.sup,b.sup;
```

PostgreSQLでは、配列型を使う方法もあります。
まず、with句で、supごとのpartの配列を求めた仮想表tmpを作成してます。
次に、supが自分より大きいこと、partの配列の件数が等しいこと、partの配列に包含関係があること
を条件として自己内部結合させてます。

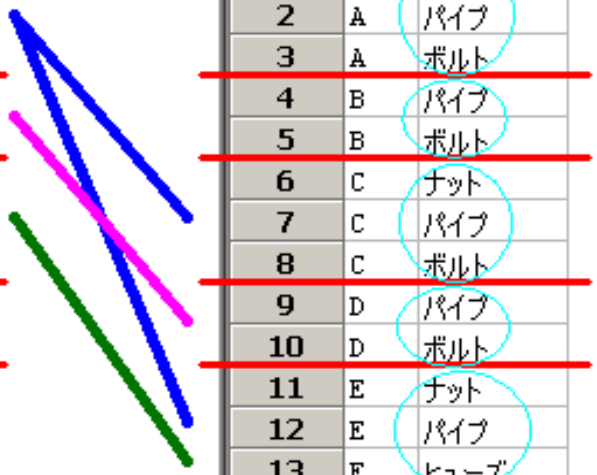
SQLのイメージは下記となります。group by supで赤線を引いて、
array_agg関数で、ベン図をイメージしてから、
仮想表tmp同士の、supが自分より大きいこと、partの配列の件数が等しいこと、partの配列に包含関係があること
を条件とした自己内部結合をイメージしてます。

tmp a

	sup text	part text
1	A	ナット
2	A	パイプ
3	A	ボルト
4	B	パイプ
5	B	ボルト
6	C	ナット
7	C	パイプ
8	C	ボルト
9	D	パイプ
10	D	ボルト
11	E	ナット
12	E	パイプ
13	E	ヒューズ
14	F	ヒューズ

tmp b

	sup text	part text
1	A	ナット
2	A	パイプ
3	A	ボルト
4	B	パイプ
5	B	ボルト
6	C	ナット
7	C	パイプ
8	C	ボルト
9	D	パイプ
10	D	ボルト
11	E	ナット
12	E	パイプ
13	E	ヒューズ
14	F	ヒューズ



```
-- ■■■ 解3 配列型を使う方法2 ■■■
with tmp as(
select sup,array_agg(part) as ArrPart
  from SupParts
group by sup)
select a.sup,b.sup
  from tmp a Join tmp b
    on a.sup < b.sup
   and a.ArrPart <@ b.ArrPart
   and a.ArrPart @> b.ArrPart
order by a.sup,b.sup;
```

partの配列同士に、互いに包含関係があることをチェックしてもいいです。
(A ⊆ B) かつ (A ⊇ B) ⇔ (A = B)
だからです。

```
-- ■■■ 解4 配列型を使う方法3 ■■■
with tmp as(
select sup,array_agg(part order by part) as ArrPart
  from SupParts
group by sup)
select a.sup,b.sup
  from tmp a Join tmp b
    on a.sup < b.sup
   and a.ArrPart = b.ArrPart
order by a.sup,b.sup;
```

array_agg関数にOrderByを指定しておいて、
配列同士が等しいかをチェックしてもいいです。

48. 非グループ化

Inventoryテーブル

goods	pieces
AAAAA	0
BBBBB	1
CCCCC	2
DDDDD	3

商品(piece)1個につき、1行として出力する。

出力結果

goods	piece
BBBBB	1
CCCCC	1
CCCCC	1
DDDDD	1
DDDDD	1
DDDDD	1

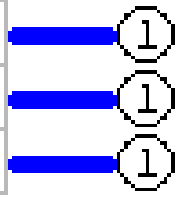
```
-- ■■■ データ作成スクリプト ■■■
create table Inventory(
goods  text primary key,
pieces integer);

insert into Inventory values('AAAAA',0),
                             ('BBBBB',1),
                             ('CCCCC',2),
                             ('DDDDD',3);
```

```
-- ■■■ 解1 再帰SQLを使う方法 ■■■
with recursive rec(goods,pieces,cnt) as(
select goods,pieces,1
  from Inventory
 where 1 <= pieces
union all
select goods,pieces,cnt+1
  from rec
 where cnt+1 <= pieces)
select goods,1 as piece
  from rec
order by goods;
```

非再帰項でpiecesが1以上である行を抽出して、
再帰項でcntのインクリメントを繰り返してます。

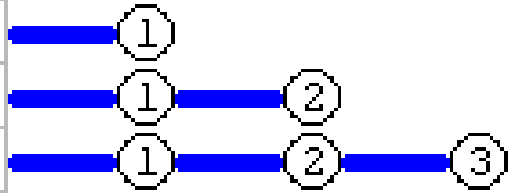
SQLのイメージ(第1段階)は、下記となります。
非再帰項による木の根の作成をイメージしてます。

	goods text	pieces integer
1	AAAAA	0
2	BBBBB	1
3	CCCCC	2
4	DDDDD	3



SQLのイメージ(第2段階)は、下記となります。
再帰項による木のノードの作成をイメージしてます。

	goods text	pieces integer
1	AAAAA	0
2	BBBBB	1
3	CCCCC	2
4	DDDDD	3



再帰SQLの処理を脳内でイメージする際には、
非再帰項と再帰項で2段階に分けてイメージすると分かりやすいです。

```
-- ■■■ 解2 Generate_Series関数を使う方法 ■■■
select a.goods,1 as piece
  from Inventory a,Generate_Series(1,a.pieces) b
order by a.goods;
```

左相関で、Generate_Series関数を使う方法もあります。

53. テーブルを列ごとに折りたたむ

Foobarテーブル

lvl	color	length	width	hgt
1	RED	8	10	12
2	null	null	null	20
3	null	9	25	25
4	BLUE	null	67	null
5	GRAY	null	null	null

lvlの降順に、各列の値を見ていって、
それぞれの最初の非null値をまとめて出力する。

出力結果

color	length	width	hgt
GRAY	9	67	25

```
-- ■■■ データ作成スクリプト ■■■
create table Foobar(
lvl    integer primary key,
color  text,
length integer,
width  integer,
hgt    integer);

insert into Foobar values(1,'RED' ,   8,  10,  12),
                         (2,null  ,null,null,  20),
                         (3,null  ,   9,  25,  25),
                         (4,'BLUE',null,  67,null),
                         (5,'GRAY',null,null,null);
```

```
-- ■■■ 解1 非nullな最大のソートキーを求める方法 ■■■
with tmp as(
select lvl,color,length,width,hgt,
max(case when color  is not null then lvl end) over() as colorTarget,
max(case when length is not null then lvl end) over() as lengthTarget,
max(case when width  is not null then lvl end) over() as widthTarget,
max(case when hgt    is not null then lvl end) over() as hgtTarget
  from Foobar)
select max(case lvl when colorTarget  then color  end) as color,
       max(case lvl when lengthTarget then length end) as length,
       max(case lvl when widthTarget  then width  end) as width,
       max(case lvl when hgtTarget    then hgt    end) as hgt
  from tmp;
```

まず、Window関数のmax関数で、colorやlengthなどが非nullであるlvlの最大値を求めてます。
次に、集約関数のmax関数で、そのlvlの行の、colorやlengthなどの値を求めてます。

SQLのイメージは下記となります。
4つの、Window関数のmax関数にそれぞれ対応した、黄緑線と青線と緑線と紫線を引いてます。

```
select lvl,color,length,width,hgt,
max(case when color  is not null then lvl end) over() as "colorTarget",
max(case when length is not null then lvl end) over() as "lengthTarget",
max(case when width  is not null then lvl end) over() as "widthTarget",
max(case when hgt    is not null then lvl end) over() as "hgtTarget"
  from Foobar;
```

出力ビュー

データの出力 | 解釈 | メッセージ | ヒストリー

	lvl integer	color text	length integer	width integer	hgt integer	colorTarget integer	lengthTarget integer	widthTarget integer	hgtTarget integer
1	1	RED	8	10	12	5	3	4	3
2	2				20	5	3	4	3
3	3		9	25	25	5	3	4	3
4	4	BLUE		67		5	3	4	3
5	5	GRAY				5	3	4	3

```
-- ■■■ 解2 array_agg関数を使う方法 ■■■
select
(array_agg(color  order by color  is null,lvl desc))[1] as color,
(array_agg(length order by length is null,lvl desc))[1] as length,
(array_agg(width  order by width  is null,lvl desc))[1] as width,
(array_agg(hgt    order by hgt    is null,lvl desc))[1] as hgt
  from Foobar;
```

count関数やmax関数やstring_agg関数などの集約関数はnullを無視して集約しますが、
array_agg関数はnullを無視しないので、
array_agg関数のOrderBy句の第1ソートキーでis null述語を使ってます。
そして、第2ソートキーをlvlの降順としてソートし、配列の要素の1番目を取得してます。

56. ホテルの部屋番号

Hotelテーブル

floor_nbr	room_nbr
1	null
1	null
1	null
2	null
2	null
3	null

room_nbrを(floor_nbr*100) + 1から始まる連番に更新する。

更新結果

floor_nbr	room_nbr
1	101
1	102
1	103
2	201
2	202
3	301

```
-- ■■■ データ作成スクリプト ■■■
create table Hotel(
floor_nbr integer,
room_nbr  integer);

insert into Hotel values(1,null),
                         (1,null),
                         (1,null),
                         (2,null),
                         (2,null),
                         (3,null);
```

```
-- ■■■ 解1 インラインビューと結合させる方法 ■■■
update Hotel a
   set room_nbr = tmp.NewVal
from (select b.ctid,
      b.floor_nbr * 100 + Row_Number() over(partition by b.floor_nbr) as NewVal
        from Hotel b) tmp
where a.ctid = tmp.ctid;
```

Window関数を使用したビューに対してUpdate文を実行すると
PostgreSQL9.3では文法エラーになりますので、
Window関数を使用したインラインビューと結合させてます。
主キーがないのでctidを使って結合してます。

SQLのイメージは下記となります。
Row_Number() over(partition by floor_nbr)に対応する赤線と黄緑線を引いてます。

```
select ctid,floor_nbr,
floor_nbr * 100 + Row_Number() over(partition by floor_nbr) as "NewVal"
  from Hotel;
```

出力ビュー

データの出力 | 解釈 | メッセージ | ヒストリー

	ctid tid	floor_nbr integer	NewVal bigint
1	(0,1)	1	101
2	(0,2)	1	102
3	(0,3)	1	103
4	(0,4)	2	201
5	(0,5)	2	202
6	(0,6)	3	301

```
-- ■■■ 解2 更新可能なwith句を使う方法 ■■■
with tmp as(delete from Hotel Returning floor_nbr)
insert into Hotel
select floor_nbr,
floor_nbr * 100 + Row_Number() over(partition by floor_nbr)
  from tmp;
```

更新可能なwith句で、Delete文とInsert文を使う方法もあります。

59. 期間を結合する

TimeSheetsテーブル

TaskID	StartDate	EndDate
1	2017-01-01	2017-01-03
2	2017-01-02	2017-01-04
3	2017-01-04	2017-01-05
4	2017-01-06	2017-01-09
5	2017-01-09	2017-01-09
6	2017-01-09	2017-01-09
7	2017-01-12	2017-01-15
8	2017-01-13	2017-01-14
9	2017-01-14	2017-01-14
10	2017-01-17	2017-01-17
11	2017-02-01	2017-02-05
12	2017-02-03	2017-02-28
13	2017-02-07	2017-02-11
14	2017-03-01	2017-03-31
15	2017-03-01	2017-03-15

TimeSheetsテーブルの重複した期間をまとめて、
期間の開始と終了を表示する。

出力結果

StartDate	EndDate
2017-01-01	2017-01-05
2017-01-06	2017-01-09
2017-01-12	2017-01-15
2017-01-17	2017-01-17
2017-02-01	2017-02-28
2017-03-01	2017-03-31

```
-- ■■■ データ作成スクリプト ■■■
create table TimeSheets(
TaskID    integer primary key,
StartDate date,
EndDate   date);

insert into TimeSheets
values( 1,date '2017-01-01',date '2017-01-03'),
      ( 2,date '2017-01-02',date '2017-01-04'),
      ( 3,date '2017-01-04',date '2017-01-05'),
      ( 4,date '2017-01-06',date '2017-01-09'),
      ( 5,date '2017-01-09',date '2017-01-09'),
      ( 6,date '2017-01-09',date '2017-01-09'),
      ( 7,date '2017-01-12',date '2017-01-15'),
      ( 8,date '2017-01-13',date '2017-01-14'),
      ( 9,date '2017-01-14',date '2017-01-14'),
      (10,date '2017-01-17',date '2017-01-17'),
      (11,date '2017-02-01',date '2017-02-05'),
      (12,date '2017-02-03',date '2017-02-28'),
      (13,date '2017-02-07',date '2017-02-11'),
      (14,date '2017-03-01',date '2017-03-31'),
      (15,date '2017-03-01',date '2017-03-15');
```

```
-- ■■■ 解1 下界との連結有無を求め、累計でグループ化する方法 ■■■
with tmp1 as(
select StartDate,max(EndDate) as EndDate,
case when StartDate
       <= max(max(EndDate)) over(order by StartDate
                                 Rows between Unbounded Preceding
                                          and 1 Preceding)
     then 0 else 1 end as WillSum
  from TimeSheets
group by StartDate),
tmp2 as(
select StartDate,EndDate,
sum(WillSum) over(order by StartDate) as GID
  from tmp1)
select min(StartDate) as StartDate,max(EndDate) as EndDate
  from tmp2
group by GID
order by GID;
```

最初に、StartDateの重複排除で、
group by StartDateとし、EndDateは、max(EndDate)とします。

期間の連結について、考察してみます。
行Xの期間が他の行Yの期間と連結するのは、下記の2つの場合が考えられます。

場合1 X.StartDate < Y.StartDate かつ Y.StartDate <= X.EndDate
場合2 X.StartDate > Y.StartDate かつ X.StartDate >= Y.EndDate

場合1と場合2の数式で、
行Aから見て他の行Bと連結しているなら
行Bから見ても行Aと連結していると分かります。

以上により
StartDateの昇順に行を見ていって、
StartDateが自分未満の行(下界の行)と連結してたら0、連結してなかったら1
とする数の累計で、グループ化すればいいと分かります。

SQLのイメージ(第1段階)は下記となります。group by StartDateで赤線を引いてます。

	TaskID integer	StartDate date	EndDate date
1	1	2017-01-01	2017-01-03
2	2	2017-01-02	2017-01-04
3	3	2017-01-04	2017-01-05
4	4	2017-01-06	2017-01-09
5	5	2017-01-09	2017-01-09
6	6	2017-01-09	2017-01-09
7	7	2017-01-12	2017-01-15
8	8	2017-01-13	2017-01-14
9	9	2017-01-14	2017-01-14
10	10	2017-01-17	2017-01-17
11	11	2017-02-01	2017-02-05
12	12	2017-02-03	2017-02-28
13	13	2017-02-07	2017-02-11
14	14	2017-03-01	2017-03-31
15	15	2017-03-01	2017-03-15

SQLのイメージ(第2段階)は下記となります。
max(max(EndDate)) over(order by StartDate
Rows between Unbounded Preceding
and 1 Preceding)
に対応する黄緑線を引いてます。

```
select StartDate,max(EndDate) as "EndDate",
max(max(EndDate)) over(order by StartDate
                       Rows between Unbounded Preceding
                                and 1 Preceding) as "Max_1_Preceding",
case when StartDate
       <= max(max(EndDate)) over(order by StartDate
                                 Rows between Unbounded Preceding
                                          and 1 Preceding)
     then 0 else 1 end as "WillSum"
  from TimeSheets
group by StartDate;
```

出力ビュー

データの出力 | 解釈 | メッセージ | ヒストリー

	startdate date	EndDate date	Max_1_Preceding date	WillSum integer
1	2017-01-01	2017-01-03		1
2	2017-01-02	2017-01-04	2017-01-03	0
3	2017-01-04	2017-01-05	2017-01-04	0
4	2017-01-06	2017-01-09	2017-01-05	1
5	2017-01-09	2017-01-09	2017-01-09	0
6	2017-01-12	2017-01-15	2017-01-09	1
7	2017-01-13	2017-01-14	2017-01-15	0
8	2017-01-14	2017-01-14	2017-01-15	0
9	2017-01-17	2017-01-17	2017-01-15	1
10	2017-02-01	2017-02-05	2017-01-17	1
11	2017-02-03	2017-02-28	2017-02-05	0
12	2017-02-07	2017-02-11	2017-02-28	0
13	2017-03-01	2017-03-31	2017-02-28	1

SQLのイメージ(第3段階)は下記となります。
sum(WillSum) over(order by StartDate) に対応する黄緑線と
group by GID に対応する赤線を引いてます。

```
with tmp1 as(
select StartDate,max(EndDate) as EndDate,
case when StartDate
       <= max(max(EndDate)) over(order by StartDate
                                 Rows between Unbounded Preceding
                                          and 1 Preceding)
     then 0 else 1 end as WillSum
  from TimeSheets
group by StartDate)
select StartDate,EndDate,WillSum,
sum(WillSum) over(order by StartDate) as "GID"
  from tmp1;
```

出力ビュー

データの出力 | 解釈 | メッセージ | ヒストリー

	startdate date	enddate date	willsum integer	GID bigint
1	2017-01-01	2017-01-03	1	1
2	2017-01-02	2017-01-04	0	1
3	2017-01-04	2017-01-05	0	1
4	2017-01-06	2017-01-09	1	2
5	2017-01-09	2017-01-09	0	2
6	2017-01-12	2017-01-15	1	3
7	2017-01-13	2017-01-14	0	3
8	2017-01-14	2017-01-14	0	3
9	2017-01-17	2017-01-17	1	4
10	2017-02-01	2017-02-05	1	5
11	2017-02-03	2017-02-28	0	5
12	2017-02-07	2017-02-11	0	5
13	2017-03-01	2017-03-31	1	6

62. レポートの整形

Namesテーブル
name

Al
Ben
Charlie
David
Ed
Frank
Greg
Howard
Ida
Joe
Ken
Larry
Mike
Neal

Namesテーブルのname列を
3行ごとに、1行で表示する(最終行の余った列は、nullとする)
出力順は、行も列も、Namesテーブルのname列の昇順とする。

出力結果

name1	name2	name3
Al	Ben	Charlie
David	Ed	Frank
Greg	Howard	Ida
Joe	Ken	Larry
Mike	Neal	null

```
-- ■■■ データ作成スクリプト ■■■
create table Names(name text primary key);

insert into Names values('Al'),
                         ('Ben'),
                         ('Charlie'),
                         ('David'),
                         ('Ed'),
                         ('Frank'),
                         ('Greg'),
                         ('Howard'),
                         ('Ida'),
                         ('Joe'),
                         ('Ken'),
                         ('Larry'),
                         ('Mike'),
                         ('Neal');
```

```
-- ■■■ 解1 商と余りで分類する方法 ■■■
with tmp as(
select name,
-1 + Row_Number() over(order by name) as rn
  from Names)
select
max(case rn%3 when 0 then name end) as name1,
max(case rn%3 when 1 then name end) as name2,
max(case rn%3 when 2 then name end) as name3
  from tmp
group by rn/3
order by rn/3;
```

0以上の整数を3で割った時の、商と余りは下記となります。

```
0 割る 3 = 0 余り 0
1 割る 3 = 0 余り 1
2 割る 3 = 0 余り 2
3 割る 3 = 1 余り 0
4 割る 3 = 1 余り 1
5 割る 3 = 1 余り 2
6 割る 3 = 2 余り 0
7 割る 3 = 2 余り 1
8 割る 3 = 2 余り 2
省略
```

以上により、商でグループ化しつつ、
余りに対応したname列を、max関数とcase式を組み合わせて表示すればいいと分かります。

SQLのイメージは下記となります。group by rn/3 で赤線を引いてます。

	name text	rn bigint	rn/3 bigint	rn%3 bigint
1	Al	0	0	0
2	Ben	1	0	1
3	Charlie	2	0	2
4	David	3	1	0
5	Ed	4	1	1
6	Frank	5	1	2
7	Greg	6	2	0
8	Howard	7	2	1
9	Ida	8	2	2
10	Joe	9	3	0
11	Ken	10	3	1
12	Larry	11	3	2
13	Mike	12	4	0
14	Neal	13	4	1

63. 連続的なグルーピング

Tテーブル

num	data
1	aaaa
2	aaaa
3	bbbb
6	bbbb
8	aaaa
10	cccc
12	bbbb
15	bbbb

各data値がnumの何番から何番まで連続しているかを、
その出現順でまとめる。

出力結果

Low	High	data
1	2	aaaa
3	6	bbbb
8	8	aaaa
10	10	cccc
12	15	bbbb

```
-- ■■■ データ作成スクリプト ■■■
create table T(
num  integer primary key,
data text);

insert into T values( 1,'aaaa'),
                    ( 2,'aaaa'),
                    ( 3,'bbbb'),
                    ( 6,'bbbb'),
                    ( 8,'aaaa'),
                    (10,'cccc'),
                    (12,'bbbb'),
                    (15,'bbbb');
```

```
-- ■■■ 解1 Lag関数でシーケンス開始を検知する方法 ■■■
with tmp1 as(
select num,data,
case when data = Lag(data) over(order by num)
     then 0 else 1 end as WillSum
  from T),
tmp2 as(
select num,data,
sum(WillSum) over(order by num) as GID
  from tmp1)
select min(num) as low,max(num) as high,data
  from tmp2
group by GID,data
order by GID;
```

numの昇順で行を見ていって、
Lag関数を使って、シーケンス開始なら1、シーケンス開始でなければ0
とする数を求め、その累計でグループ化してます。

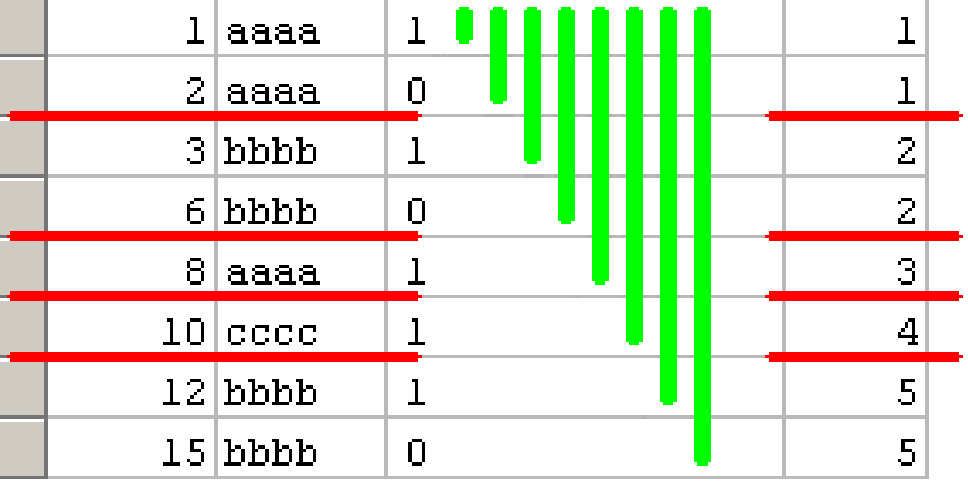
SQLのイメージは下記となります。
sum(WillSum) over(order by num) as GIDに対応する黄緑線と、
group by GID,dataに対応する赤線を引いてます。

```
with tmp1 as(
select num,data,
case when data = Lag(data) over(order by num)
     then 0 else 1 end as WillSum
  from T)
select num,data,WillSum,
sum(WillSum) over(order by num) as "GID"
  from tmp1;
```

出力ビュー

データの出力 | 解釈 | メッセージ | ヒストリー

	num integer	data text	willsum integer	GID bigint
1	1	aaaa	1	1
2	2	aaaa	0	1
3	3	bbbb	1	2
4	6	bbbb	0	2
5	8	aaaa	1	3
6	10	cccc	1	4
7	12	bbbb	1	5
8	15	bbbb	0	5



```
-- ■■■ 解2 旅人算メソッドを使う方法 ■■■
with tmp as(
select num,data,
 Row_Number() over(order by num)
-Row_Number() over(partition by data order by num) as distance
  from T)
select min(num) as low,max(num) as high,data
  from tmp
group by data,distance
order by min(num);
```

中学受験の算数で有名な旅人算の感覚を使う方法もあります。

1進む条件が異なる4人の旅人(旅人X,旅人A,旅人B,旅人C)が
数直線の原点からプラス方向に同時にスタートしたとして、
●必ず1進む旅人Xの位置 = Row_Number() over(order by num)
●dataがaaaaなら1進む旅人Aの位置 = Row_Number() over(partition by data order by num)
●dataがbbbbなら1進む旅人Bの位置 = Row_Number() over(partition by data order by num)
●dataがccccなら1進む旅人Cの位置 = Row_Number() over(partition by data order by num)
と考えてます。

そして、group by data,distance によって、
dataに対応した旅人の種類(旅人A,旅人B,旅人Cのどれか)と、
旅人Xとの距離でグループ化してます。

SQLのイメージは下記となります。
Row_Number() over(order by num)に対応する1人の旅人と、
Row_Number() over(partition by data order by num)に対応する3人の旅人をイメージし、
group by data,distanceに対応する赤線を引いてます。

```
with tmp as(
select num,data,
Row_Number() over(order by num) as "TabibitoX",
Row_Number() over(partition by data order by num) as "TabibitoA,B,C"
  from T)
select num,data,"TabibitoX","TabibitoA,B,C",
"TabibitoX" - "TabibitoA,B,C" as distance
  from tmp
order by num;
```

出力ビュー

データの出力 | 解釈 | メッセージ | ヒストリー

	num integer	data text	TabibitoX bigint	TabibitoA,B,C bigint	distance bigint
1	1	aaaa	1	1	0
2	2	aaaa	2	2	0
3	3	bbbb	3	1	2
4	6	bbbb	4	2	2
5	8	aaaa	5	3	2
6	10	cccc	6	1	5
7	12	bbbb	7	3	4
8	15	bbbb	8	4	4

66. 数独パズル

ニコリの数独の問題を解きます。

問題

8					5	1		
		1				8		
	4		2				9	
				3				2
1	2	3	4		6	7	8	9
6				1				
	8				9		5	
		2				4		
		7	6					1

答え

8	3	9	7	6	5	1	2	4
2	6	1	3	9	4	8	7	5
7	4	5	2	8	1	3	9	6
5	9	4	8	3	7	6	1	2
1	2	3	4	5	6	7	8	9
6	7	8	9	1	2	5	4	3
3	8	6	1	4	9	2	5	7
9	1	2	5	7	3	4	6	8
4	5	7	6	2	8	9	3	1

```
-- ■■■ 数独を解くのに使う文字列関数 ■■■
select
StrPos(Val,'a') as tes1,
StrPos(Val,'X') as tes2,
SubStr(Val,1,2) as tes3,
SubStr(Val,3,2) as tes4,
OverLay(Val placing 'XYZ' from 2) as tes5,
OverLay(Val placing 'XYZ' from 2 for 0) as tes6
  from (values('abcdef')) as Work(Val);
```

出力結果

tes1	tes2	tes3	tes4	tes5	tes6
1	0	ab	cd	aXYZef	aXYZbcdef

数独を解くのに使う文字列関数の紹介です。

```
-- ■■■ 解1 幅優先探索を使う方法 ■■■
with recursive Question(Val) as(
values('800005100'
    || '001000800'
    || '040200090'
    || '000030002'
    || '123406789'
    || '600010000'
    || '080009050'
    || '002000400'
    || '007600001')),
Renban1To9 as(
select Cnt
  from generate_series(1,9) as Renban1To9(Cnt)),
rec(LV,Val) as(
select 1,Val from Question
union all
select LV+1,
case when SubStr(Val,LV,1) = '0'
     then OverLay(Val placing Cnt::text from LV)
     else Val end
  from rec a Join Renban1To9 b
    on a.LV <= 81
   and (SubStr(a.Val,a.LV,1) = '0' or b.Cnt=1)
 where SubStr(Val,LV,1) != '0'
    or (StrPos(SubStr(Val,(LV-1)/9*9+1 ,9),Cnt::text) = 0 --横チェック
    and StrPos(SubStr(Val,(LV-1)%9+1   ,1),Cnt::text) = 0 --縦チェック
    and StrPos(SubStr(Val,(LV-1)%9+1+ 9,1),Cnt::text) = 0
    and StrPos(SubStr(Val,(LV-1)%9+1+18,1),Cnt::text) = 0
    and StrPos(SubStr(Val,(LV-1)%9+1+27,1),Cnt::text) = 0
    and StrPos(SubStr(Val,(LV-1)%9+1+36,1),Cnt::text) = 0
    and StrPos(SubStr(Val,(LV-1)%9+1+45,1),Cnt::text) = 0
    and StrPos(SubStr(Val,(LV-1)%9+1+54,1),Cnt::text) = 0
    and StrPos(SubStr(Val,(LV-1)%9+1+63,1),Cnt::text) = 0
    and StrPos(SubStr(Val,(LV-1)%9+1+72,1),Cnt::text) = 0
    and StrPos(SubStr(Val,((LV-1)%9)/3*3+1+9*(((LV-1)/9)/3*3)   ,3),Cnt::text) = 0 --正方形チェック
    and StrPos(SubStr(Val,((LV-1)%9)/3*3+1+9*(((LV-1)/9)/3*3)+ 9,3),Cnt::text) = 0
    and StrPos(SubStr(Val,((LV-1)%9)/3*3+1+9*(((LV-1)/9)/3*3)+18,3),Cnt::text) = 0))
select SubStr(a.Val,1+9*(b.Cnt-1),9) as Answer
  from rec a,Renban1To9 b
 where a.LV=82
order by b.Cnt;
```

出力結果
Answer

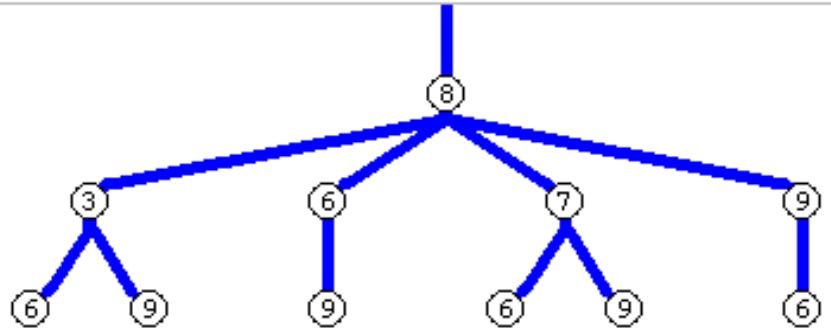
839765124
261394875
745281396
594837612
123456789
678912543
386149257
912573468
457628931

再帰項で、1から9までの連番テーブルとの内部結合を繰り返しつつ、
Where句で、その連番でマスを埋めることが可能かをチェックしてます。

SQLのイメージ(探索木の深さは3まで)は下記となります。幅優先探索の探索木をイメージしてます。
未確定マスは、1から9までのノードが作成候補になりつつ、枝切りされていくイメージです。

高さ1のノードが、問題の1番上の行の1番左のマス
高さ2のノードが、問題の1番上の行の左から2番目のマス
高さ3のノードが、問題の1番上の行の左から3番目のマス
に対応します。

	Val text
1	800005100001000800040200090000030002123406789600010000080009050002000400007600001



第2部 参考リソース

書籍

達人に学ぶ SQL徹底指南書

Web

『SQLパズル 第2版』サポートページ

次回予告

需要があって、いつか私の気が向いたら、
どこかで講演させていただくかも？？？

- OracleSQLクイズ(翔泳社)の問題を解く　(Oracle)
- SQLクックブック(オライリー)の問題を解く　(PostgreSQLかOracle)
- SQL徹底指南書(翔泳社)の後半問題を解く　(PostgreSQLかOracle)
- Joe Celko's SQL Puzzles and Answersを意識して、
  明智版SQLパズル　(PostgreSQLかOracle)

- 詳説正規表現のPostgreSQLの章
- 詳説正規表現のOracleの章

質疑応答

私が回答できない質問は、誰か代わりに回答して下さい :-)
休憩時間や懇親会(多分参加)でも質問受け付けます。